Supercomputing Pipeline for The Ark (SPARK)

Thilina Sunimal Ranaweera

Submitted in total fulfilment of the requirements of the degree of

Doctor of Philosophy

Melbourne School of Population and Global Health
The University of Melbourne

August 2018

Produced on archival quality paper.

Abstract

Data is the primary asset of biomedical researchers and the engine for both discovery and translation. The challenge is to facilitate the discovery of new medical knowledge, supported by advanced computing technologies based on heterogeneous medical datasets. At the beginning of the 21st century, the much-anticipated Human Genome Project was completed, thus commencing an era of high-dimensional genomic datasets. Additionally, the internet era has introduced advanced and more user-friendly computing technologies for data storage, processing, and analysis. The biomedical informatics research field has emerged to fill the void between medical research and computing technologies.

The Ark is an open-source web-based medical research data management platform which currently supports multiple medical research studies. The Ark's software architecture is modular, capable of managing multiple studies and sub-studies in a single database, including laboratory information and questionnaire data. The Ark's role-based security infrastructure provides a secure environment for biomedical datasets. Prior to the work presented in this thesis, The Ark did not have the capability to effectively manage or analyse genomic data, nor model pedigree structures.

This thesis presents a next-generation biomedical data management system, the Supercomputer Pipeline for The Ark (SPARK). SPARK provides a novel data management approach to pedigree and genomic datasets. The SPARK project en-

ables access to high-performance computing (HPC) using a software architecture that builds upon The Ark's researcher-friendly web-based interface. The Ark genomics module can accommodate multiple HPC facilities simultaneously, enabling multiple SPARK nodes that are attached to each HPC facility to operate independently. A SPARK plugin format for analysis algorithms is implemented, promoting the exchange of algorithm implementations within the global genomics community. In addition to HPC support, the genomics module provides functions to manage high-dimensional genome-wide association study (GWAS) data. A mechanism to manage and analyse GWAS datasets using a mixed relational/non-relational model is designed and implemented.

A further major contribution of this thesis is the first open-source pedigree data modelling and visualisation tool that is integrated with a sophisticated medical data management system. The Ark's new pedigree module is capable of modelling n-th degree relatives and twins, including multiples. It can dynamically infer family structures based upon parental and twin relationships alone, and can visualize these structures with configurable annotations, for example, affected status.

In summary, SPARK implements a novel knowledge discovery platform that incorporates HPC and genomic data management capabilities into The Ark. SPARK empowers medical researchers, who do not have a computer science background, to exploit HPC resources for genomic medical research, share algorithms and data, and extract results in a user-friendly way. The pedigree module extends The Ark to handle family and twin-based studies, which are a powerful tool to better understand the origins of disease. SPARK represents an important milestone in translating HPC and Big Data management technologies to the medical research community. The design and implementation described in this thesis are sufficiently generic that they could be readily extended to other medical domains involving Big Data, and analyses that demand HPC.

Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Thilina Sunimal Ranaweera, August 2018

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my primary supervisor, Dr Adrian Bickerstaffe for his continuous support, guidance and inspiration throughout my PhD journey. Thank you for your understanding, patience, bestowing your knowledge and guiding me to the right research direction, and enriching my life with your advice and words of wisdom. I could not have imagined a better supervisor for my PhD candidature.

Besides my primary supervisor, heartfelt thanks go to my co-supervisors Dr Enes Makalic and Professor John L. Hopper. Thank you for your encouragement, insightful discussions, providing guidance and feedback to enrich my research immensely. Specially, I would like to thank Professor John L. Hopper for providing the opportunity to work in his research group and supporting my research work from his funds. I would also like to thank my fellow researchers and work colleagues at the Centre for Epidemiology and Biostatistics, The University of Melbourne School of Population and Global Health, for timely help and insightful discussions.

I have met some amazing people during my PhD candidature, especially my colleagues at the Centre for Genetic Origins of Health and Disease, The University of Western Australia. Specially, I would like to thank Professor Eric Moses on initiating open-source based The Ark project to manage the biomedical research datasets, which has become the foundation of my PhD research. Also, I would like to thank The Ark development team based on the Centre for Genetic Origins

of Health and Disease, The University of Western Australia for their effort and dedication to this project. Thank you for making my life enjoyable and for many unforgettable experiences with The Ark developments.

Also, I would like to thank Dr Gillian Dite and Dr Julie Cantrill for proofreading my thesis and providing suggestions for improvements. Their suggestions have helped to improve my thesis content immensely.

PhD journey is enriched with achievements and constant challenges, and I was very fortunate to have my wife by my side taking the same ride. My dearest wife Nilu, thank you for all your understanding, supporting me during difficult times, taking care of me and being patient with me when I was too busy to manage responsibilities as a husband. Also, thank you for making me laugh and making my life enjoyable. My PhD journey has become extra special as I had the greatest gift of my life, my daughter Minuli during my candidature. She made my life super busy but changed it for the better and motivated me to be successful. I appreciate my little girl for abiding my ignorance and the patience she showed during my thesis writing. Please accept sincere apologies of your Thaththa for not being able to spend more time and play with you.

Without my mother, father, sister, and brother I would not be who I am today, and I am very fortunate to have them by my side. My mother supported me, motivated me and stayed with me like my own shadow during good times and hard times. Thank you Amma for being so special, kind, understanding and always wishing the best for me. My father always guided me through his knowledge and encouraged me to pursue higher studies. Thank you Taththa for believing in me and guiding me in the right direction.

Finally, I would like to thank everyone who has been a friend to me, thank you very much for your friendship and the good times we shared.

To my beloved family and respected teachers

Contents

List of Figures

List of Tables

A novel web-based software system to accelerate advances in medical research aspects by implementing a knowledge discovery platform that enables HPC and Big Data management for heterogeneous medical datasets

1 Introduction

1.1 Biomedical informatics

Data is a major component of medical research studies, and data-related requirements vary widely according to a study's research objectives and researcher capabilities and expertise. History provides important evidence of how data has been used for major medical research breakthroughs. For example, an early demonstration of data-driven medical research dates back to the 17^{th} century, when a vitamin C deficiency among sailors was discovered by analysing naval dietary records [8]. In the mid-19^{th} century, dot maps were used by Dr John Snow to discover the water resources causing the cholera epidemic [9]. These are some examples of scenarios which emphasise the importance of using data for medical discoveries with

rudimentary analysis tools – pen and paper.

The 20th century has witnessed a wide expansion of technologies and inventions in the scientific research domain. In particular, large advances have been made in the science and engineering fields, with levels of understanding boosted by technological advances and research findings in the last century. There were two major scientific breakthroughs in the 20th century which have had a significant impact on the present day biomedical research domain. The first breakthrough was in the 1950s when scientists discovered the basic element of living organisms, deoxyribonucleic acid (DNA) [10]. The second breakthrough was the 1960s invention of the solid-state transistor which has significantly reduced the size of electronic devices and decreased the cost of building computationally powerful, compact electronic devices [11]. Valve-based computers existed before the invention of solid-state transistors, but the latter took a giant leap forward in reliability, speed and density of these switching units.

Discovery of DNA has opened a whole new research pathway in biology which is specialised in finding the heritability material in living organisms. The newly discovered DNA structure has led to further investigation on the causality of functional biology, and its hereditary effects passed through the generations [12]. Increased technological innovations have provided the necessary infrastructure support for further analysis of biological information in advanced laboratory facilities. In recent decades, the technological innovations have led to discovering the protein structures. Furthermore, discovered ribonucleic acid (RNA) which transfer the information inside a cell, and genes which represent the protein-encodings in the chromosome regions.

Computers have evolved rapidly after the introduction of the transistor to electronic device manufacturing. Computer size has rapidly decreased and the power of a single computer has significantly increased with the introduction of advanced electronic manufacturing techniques. By Moore's law [13], the density of transistors able to be placed on a processing unit has approximately doubled every two years,

producing rapid leaps in the processing power of computer processors and integrated circuits more broadly. Similar to the computing power, the storage devices capacity has been doubled per 18 months according to Amdahl's Law [14]. This computing power and capacity has transferred directly into the personal computing space, and the growing ubiquity of desktop computers has aided the biomedical research process by attracting a much wider computer user community.

Based on the availability of computing resources for the day-to-day needs of society, most of the robust data management models and computer application architecture methodologies have been developed by computer scientists. The field of software engineering has emerged based on the objective of improving the theory and practice of computer applications. In the 1970s, a relational data model [15] was introduced to the data storage schema design and widely accepted by the software field as a common data model for a wide range of the datasets. Data schemas represent the design of the data model for a particular problem. Up until the final years of the 1990s, the relational data model dominated the data management design of the software applications. The relational data model facilitated the decoupling of the data schemas from the rest of the software application which interacts with it. This separation of data modelling and the software application layer, and the software frameworks that computer scientists have developed to implement this approach are highly flexible, capable of supporting a wide range of uses. Computer languages have been developed with explicit consideration given to support the robust implementation of software tools that separate the data modelling layer from the application layer, including the user interface. These languages include high-level languages independent of the computing systems, for example C [16], C++ [17] and Java [18], and web-specific languages operated on Hypertext Transfer Protocol, for example PHP [19] and Ruby on Rail [20].

In recent decades, advancements in the computer science have enabled the development of software systems to support biomedical research. This opportunity has led to a new research field called biomedical informatics which enhances medi-

cal research outcomes by developing novel software systems [21]. The early focus of this research field has been aimed towards developing databases which are capable of storing study datasets derived from existing research work. A number of datasets derived from medical research study cohorts contained the physical attributes of study participants and their family history, including the relatives who are connected with the study participants. Therefore, separate database schemas have been developed to store study participants' attributes (demographic information, phenotypic attributes, study information, etc.) and family relationships. Until recently, data managers make existing research datasets available for the medical researchers to analyse.

1.1.1 Genomics

The technical evolution has enabled scientists to conduct the experiments in their biomedical laboratory facilities to unravel the biological functionality of living organisms. In the 1980s, scientists fully sequenced the full genome of single cell organisms based on the available technology at that time. To investigate and understand the hereditary material of living organisms, a new biological research field called genomics emerged [22]. The linkage studies conducted based on the available genomic datasets have shown significant impact on the hereditary diseases based on genomic causalities. Cosegregate genetic markers were identified at specific chromosome locations (genes) in the Mendelian families (the family members connected with the same parental relationships) affected by a particular disease [23]. The initial finding of the BRAC1/2 gene mutation susceptibility to breast cancer is a typical example of investigating the genomic information in the linkage studies [24].

In the early 1990s, scientists started the Human Genome Project (HGP) to sequence the full human genome which was completed in early 2000 [25]. The results of the HGP encouraged scientists to explore the wide variety of human genomes further. The 1000 Genome project [26] was started to analyse human genomes

based on factors which contributed to genetic discrepancies among different human such as ethnicity, geographic distance, etc. Based on the 1000 Genome sequencing results, the HapMap project was established to map common variants in the human genome [27]. Scientists have further investigated the genetic causality of non-heritable diseases which are common in society. The decrease in the cost of genome sequencing and mapping the common variants in the human genome has enabled the sequencing of specific genomic regions in large population samples for a reasonable cost. By comparing the population-based affected (case) and unaffected (control) genomic samples, researchers were able to search for common genetic variants among the cases. This methodology is referred to as a Genome-Wide Association Study (GWAS), and after the first wave of GWAS, researchers have identified wide genomic regions associated with diseases.

This first wave of the GWAS has emphasised that complex genetic variations cause various human biological factors. Therefore, researchers decided to conduct a wider GWAS analysis rather than limiting it to a single group or population sample. As GWAS studies become larger, the datasets generated from the studies will grow rapidly. The GWAS studies have generated large phenotypic and genomic datasets from the increasing large groups of study participants. A new set of requirements has been added to the biomedical study data management systems. These present a challenge of integrating the GWAS datasets with existing biomedical data management systems which are tied to the size of the genomic datasets, often in orders of magnitude larger than phenotype datasets, privacy and security challenges, and integration with the high-performance computing (HPC) resources for required analysis.

1.2 The software for biomedical research

The introduction of the computers and software systems has added much wider capabilities to data storage and retrieving valuable information in medical research.

Medical researchers have been eager to use computer software systems to enhance the existing research capabilities from the 1960s [28]. The computer-based medical research software has been mostly limited to the data management of existing datasets [29]. The reasons for such limitations are widely attributed to the lack of computing expertise held by medical researchers, a preference for clinically based research, and lack of computing capabilities (cost effective computer hardware and advanced software systems) to cater for a wide range of medical domain-specific requirements. In particular, in the early days of the computer science there was a lack of capabilities to interpret the multi-dimensional information sources, compounded by a lack of hardware capabilities to process this information and present the results on-demand.

Emerging rapidly in the new millennium (sometimes called the "dot-com bubble"), technology companies have worked to advance World Wide Web technologies and apply these systems to commercial applications, typically described as the e-commerce application platforms in the software engineering field. Computer science has produced much-needed technological support for the e-commerce platforms (Amazon[1], eBay[2], and Facebook[3] are some of the examples), providing a theoretical foundation for their design and implementation. However, while the e-commerce domain has benefited greatly from this work, the biomedical research domain did not keep in-step with such advances.

The United States Food and Drug Administration (FDA) agency foresaw this rapid growth in the software field and its impact on the benefit of medical research [29]. In 2002, FDA produced a publication on the lack of using currently available software capabilities to fulfil the research gaps in medical science. The FDA report highlighted the lack of using cutting edge web technologies to manage the medical data repositories, software applications to diagnose the medical symptoms and predict the disease susceptibilities, and importantly, the lack of medical

[1] `http://www.amazon.com`
[2] `http://www.ebay.com`
[3] `http://www.facebook.com`

innovation following the high availability of the computing resources. The FDA report has had a significant impact to the medical research community which is now using the available software support to enhance their research work and starting to build new systems to cater specifically medical research requirements.

As an initial phase, some web-based medical data management systems have been developed, for example, The Ark [30] and REDCap [31]. Independently of these systems, software tools have been developed to manage and visualise the family relationships (i.e. pedigrees), in addition to bioinformatics tools that enable a researcher to conduct analysis based on genomic datasets. The development of bioinformatics tools includes workflow management systems managing multi-step analysis processes, HPC clusters to facilitate complex analyses, and large reference databases of genomic sequencing outputs and associated meta-information.

Biomedical data management systems fall into two major categories. One set of biomedical data management tools are designed to support the datasets of phenotype information, and others are designed for the genomic data management. The phenotypic data management are mainly divided into three major categories: (1) study consent, contact, correspondence, file attachment, and general activity data related to the medical study, (2) laboratory (biocollection/ biospecimen) data management, and (3) pedigree data management. The Ark system is an example of the phenotypic data management, with phenotype data management capabilities that are similar to comparable systems. A summary of typical phenotypic data management system requirements includes: (1) capability to manage one or more study datasets inside a single system, (2) a centralised relational database to hold multiple phenotype datasets attached to a study, and (3) user access restrictions to limit data visibility and manipulation capabilities on a per-user or per-user-group basis. These phenotypic data management systems are capable of supporting biomedical study data management for research registries and longitudinal cohort studies.

1.2.1 The Ark

Developed as a collaboration between The University of Western Australia and The University of Melbourne, The Ark is an open source web-based medical data management system implemented using Java web technologies [32] and the MySQL database system [33]. The Ark is a highly configurable system capable of handling study, subject (participant), pedigree, phenotypic and biospecimen data, in addition to tracking and invoicing research-related work requests and facilitating custom reporting and data extraction. The Ark also features a fine-grained security architecture that allows each study administrator to permit and restrict access at a role-based level. For example, a laboratory user may have limited read-only access to participant demographic information, but full read, write and configuration control within the biospecimen management module.

Genomic data management came into wider application space after completing the HGP [25]. GWAS datasets are one example of large, highly structured data. High volumes of complex data are being generated by both commercial and non-commercial, including academic, sources. These datasets are often in the order of gigabytes, terabytes and petabytes, and can often be highly dynamic, i.e. rapidly changing and growing. This class of data is referred to as Big Data, and the management of such data exceeds the capacity of existing relational-based data management systems. The emerging data trend has encouraged non-relational data models developed specially to handle datasets of Big Data magnitude. There are efforts to build sophisticated software frameworks to provide distributed data containers to manage the Big Data sets (e.g. The Google MapReduce framework [34]) and provide parallel execution environment to support computationally complex data processing algorithms (Open Message Passing Interface (MPI) [35]). Specialised software architectures have been developed to support the Big Data management and interface HPC platforms, sometimes via web interfaces (SeqWare query engine [36] and DataSHIELD analysing platform [37]). Also, developed genome browsers have become quite popular among the genomic researchers to browse through the ge-

nomic regions of living organisms in a single place and export datasets based on researcher query parameters. The Ensembl [38] and UCSC [39] are two most popular genome browsers available for genomic researchers, and they contain multiple genome datasets belonging to a number of living organisms.

1.2.2 Limitations of existing tools and systems

The existing biomedical data management systems have been faced with a substantial challenge of addressing the need to support heterogeneous medical research datasets generated by contemporary genomic studies, including GWAS. A key part of this challenge is to incorporate user-friendly interfaces into these systems, such as web-based interfaces, for medical researchers who do not have specialised computing training or expertise.

The utmost question is how to integrate standalone analysing tools and datasets into a single study space. The existing biomedical data management systems work with web-based or desktop-based user interactions and are backed by relational data management systems to store the datasets. Also, there are specialised family data management systems like Progeny [40] which store the existing relationship details and family information inside a specialised relational database schema. Much of the pedigree data modelling has been focused on developing pedigree visualisation tools rather than establishing pedigree datasets in a study-based data repository [41]. The pedigree data modelling inside a study context has not been considered while designing the RedCap or The Ark systems initial design specifications.

The enhancements in genome sequencing technologies have contributed to the generation of larger genomic datasets. These datasets have been stored in specialised systems with unique hardware and software infrastructure support for the objectives. In particular, the genome browsers are designed to store full sequence genomic datasets belonging to various species and have unique storage and access specifications for the systems. Therefore, researchers have to connect to these sys-

tems, understand the specifications and later extract the datasets based on the available user privileges. Also, dbGaP [42] type specialised data stores have been designed to store GWAS datasets which are researchers can use. One common challenge of using the above-mentioned genomic data repositories is they deviate from a single study-based system integrated with a wide spectrum of genomic data management specifications.

Third party tools are commonly used for the genomic data analysis. The PLINK toolset is a widely accepted analysing framework for the GWAS studies [43]. To use the PLINK as the analytical platform, researchers have to transform their datasets into the PLINK supporting file type manually and prepare the analytical pipelines by creating the execution scripts. The given tasks require advanced knowledge in computing systems. Similarly, there are individual programs developed by the researchers to support genomic data management and analysis which reside in individual workspaces. Therefore, centralised analytical method repositories need to be shared with the wider research community to enable easy access to the real-time environment with automated data management capabilities.

Recent advancements in sequencing technologies have increased the capability of processing datasets which are categorised into the Big Data specification with minimum cost [44]. The sequencing has been carried out independently and the data repositories provided the existing datasets. Researchers required personalised data centres to hold the extracted datasets and they conducted the experiments on selected datasets. Due to the complexity and size of the datasets expensive HPC power was required for the analysis. The SeqWare [41] and Galaxy [45] systems were built to support genomic data storage and analysing. Analysing the design and deployment instructions, the existing systems were specific to a single research facility and required certain changes for their design to interact with collaborative research facilities. Therefore, a secure data management platform capable of providing much-shared data sources and HPC power was required for the researchers. Also, it was essential to share the data sources, analysing methods,

and results among the collaborative genomic research scientists to initialise wider accepted conclusions for their research findings [46].

1.3 SPARK

At the commencement of the project described in this thesis (see Chapter 3), The Ark lacked the ability to manage pedigree and genomic data, and it did not provide a means to utilise HPC platforms for genomic data management. The Ark's original data management database schema was designed to handle multiple study datasets with multiple study participants attached to each study. Participant-related information is recorded by a set of modules that have been programmed to capture each conventional (non-genomic) study data. Using The Ark as the basis of the work presented in this thesis, the SPARK project introduces a researcher-friendly pedigree and genomic data management and analysis system that is integrated with a sophisticated biomedical data management tool.

A set of research objectives were designed to investigate the optimum method to manage pedigree and genomic datasets inside The Ark. The outcomes of these research objectives have been evaluated, with respect to comparable systems, in Chapter 7. The details of the implementation and access to the live demonstration of the SPARK is provided in Appendix A.

1.3.1 Pedigree data management

Pedigree data is a crucial element of medical research. Therefore, heredity studies have collected information on a large number of families and the available phenotypic information related to their family members. Existing databases have been developed to handle pedigree relationships (e.g. OMIM [47]) and a number of visualisation tools developed to visualise the pedigree relationships using computer graphics technologies. The existing family datasets either exist in individual

databases separate to the study-based data management systems or study-specific data management systems have been developed to accommodate datasets with individual study participants. Therefore, most of the family relationship management has been coordinated by the data managers while the researcher makes a request to export the family datasets from the databases according to the selected pedigree visualisation tool support format.

To satisfy the pedigree data management requirement within a study space, research has focused on creating: (1) a database schema to store the pedigree information, (2) a researcher-friendly web interface to record the relationships, (3) algorithms to infer the family relationships with respect to a particular subject, (4) a means to visualise pedigree structures dynamically, and (5) a means to control researcher access to pedigree functions.

Pedigree data schema store normalised family relationship data along with the study context. The data model requires only the setting of parental and twin relationships, which simplifies the task greatly, and brings more flexibility when compared to explicitly-defined relative types (see Section 4.5). A simplified web user interface gives the researchers the ability to define the family relationship along with validation capabilities (see Section 4.7). Without the involvement of the data manager, the pedigree module can automatically populate the list of inferred relatives for a subject (see Section 4.6). Visualisation is enabled via a third party pedigree visualisation tool with configurable parameters along with the study context. Also, The Ark's existing role-based permission module has provided access controls concerning the pedigree module.

Having the pedigree data management inside a study-based biomedical data management system has introduced a novel way of managing relationships. This software design has enabled researchers to self-manage pedigree data via The Ark's new pedigree module, without reliance on assistance from expert data management staff. As the pedigree module is integrated with The Ark, it adds a sophisticated new facet to the system's existing research data management capabilities. Studies

with data already managed by The Ark can now incorporate pedigree information without the need to change other aspects of the study data.

1.3.2 Access to high-performance computing resources

Computation power has become a critical component in medical research in the 21st century. Specialised computer hardware (supercomputers, computer clusters and cloud computing) has been built to provide the computation power for complex analysis of medical research data. Recent progress in the medical research and available technologies has generated increasingly large volumes of heterogeneous medical datasets, especially in the genomic research domain. Researchers who are involved in genomic research depend greatly on the HPC sources to assist with their analyses, which would otherwise require time to acquire the computing expertise to exploit these systems to their potential.

A computer platform that enables the much wider use of HPC resources and easily interacts with these facilities via a researcher-friendly web interface is something that would benefit the medical research community. There are specialised systems built to support an individual HPC facility but have thus far been unable to reach a general solution to cover a wide spectrum of the computing facilities (e.g. SeqWare Query Engine based on the Amazon E3 cloud [47], and Google genomics [48]). Biomedical data management systems are designed to act as a data repository for research datasets, supporting functions such as querying, filtering and quality control processing. Such systems often do not provide the ability to work with large heterogeneous genomic datasets, e.g. GWAS datasets, which can potentially require a significant amount of computational power to query and use as input to complex analysis algorithms.

The SPARK research aims to come up with a novel software design which accommodates wide HPC facilities for a biomedical study data analysis. There will be independent computer nodes, called SPARK nodes, initiated at the HPC facil-

ities. These SPARK nodes are web applications which operate specifically to the HPC infrastructure respecting the security parameters, access mechanisms and operational instructions. The SPARK nodes are mainly responsible for carrying out the analysis and to manage the data sources connected to them. SPARK nodes are exposing the secure web services matched with The Ark genomic module application programming interface (API). The Ark users can communicate with registered SPARK nodes and carry out the experiment based on data sources available and computing methods initiated in the facilities (see Section 5.2).

After carefully analysing existing software architecture designs, a microservice architecture has been designed and implemented [49]. The microservice approach allows The Ark to communicate with HPC sources using a generic (non-platform specific) interface. In this way, the microservice approach brings "future proofing" to SPARK, ensuring that it will be capable of interfacing new types of HPC systems as they become available. The introduction of The Ark genomics module has provided a common web interface to the Representational State Transfer (REST) Protocol-based services enabled via the SPARK nodes deployed to communicate with the HPC facilities. The microservice-based design provides an independent service implementation within the independent SPARK nodes communicating with the computing facilities on the security constraints. The Ark communicates with the SPARK nodes via secure REST-based web services. The software design introduced here provides a common mechanism to enable HPC source to exist in medical data management systems with the ability to monitor the progress of the complex computing and download the results on-demand (see Section 5.4).

1.3.3 Genomic data management

The problem of designing and implementing an efficient and capable genomic data management is highly non-trivial. The obvious choice of a relational database with one row per subject, one column per allele is not practical due to the large number of

alleles (potentially millions). This naturally leads to the "normalised" form, which involves transforming the table design such that alleles are recorded row-wise in a separate table that is linked to a table of sample information. However, this does not scale either, because a single dataset might require five million rows per sample, and there may be thousands of subjects. The problem is magnified within a multi-study system such as SPARK, as a single study may have many datasets, and combined with the datasets of other studies in the system, the number of rows in the normalised database design (schema) could enter into the billions.

The common characteristics of the genomic datasets have exceeded the capacity of popular relational database management systems by size and complexity. The relational database systems have been designed to operate with a pre-structured dataset which are usually called normalised datasets. The size of the genomic datasets is difficult for the relational databases to process because the datasets usually expand horizontally rather than the usual vertical increase in relational datasets (e.g. 5 million allele pairs for 1000 participants). Genomic datasets such as GWAS datasets are an example of Big Data, and therefore the technologies developed to address the requirements of Big Data management are relevant to SPARK. The non-relational databases or the NoSQL databases are specialised in managing the datasets which exceed the capabilities provided by the relational database management systems. NoSQL databases have specialised data models other than the relational model suitable for the domain-specific datasets (Key-Value, Document, Columnar, and Graph are the popular NoSQL data models) and the research objective is to select a suitable NoSQL database model for GWAS data management.

The SPARK research has focused on managing the GWAS datasets within a web-based genomics module for The Ark. GWAS datasets are commonly stored as "regular" files on disk within file-based data repositories. However, for a system to support researcher-driven data processing operations (e.g. querying), these files must first be brought into a database. To demonstrate that a relational database

approach to this problem is not practical, an experiment was designed to (1) show that the size-on-disk for typical GWAS datasets is impractically large, particularly for a multi-study system such as SPARK, and (2) empirically show that even basic data processing operations (e.g. loading a GWAS dataset into the database) incur impractically long running times. The first phase of the experiment results concluded that the original file structure required less disk space compared to the relational data schema. The excess physical disk space acquisition is explained by the overhead created by relational data schema properties, including table structure meta-information, links between tables ("foreign key constraints"), and inefficiency in data representation due to the range of available column types.

The same experiment was conducted with the non-relational Cassandra columnar data schema using binary encoding of allele information. The Cassandra columnar database schema can accommodate up to two billion columns per table according to the specification, hence would be able to accommodate allele samples with five million base pairs. Therefore, there is a possibility to accommodate a single allele pair per each column field.

The experiment has shown that the non-relational data schema can save more disk space at a data export rate more than ten times faster than the relational data schema. Therefore, a comparison of a simulated dataset for 1–100,000 allele samples for 1–100,000 population sample was made between relational and non-relational data management systems. The experiment was conducted in a controlled computing environment dedicated to the execution of insertion and extraction operations. An outcome of the experiment considered the insertion times of each data management system and their disk space acquisition for each dataset. The introduction of the binary encoding to represent the allele information compressed the storage of GWAS data resulting in less physical disk space allocation required in the non-relational columnar data model.

Based on the experiment results, non-relational columnar-based database schema has been implemented to load GWAS dataset into a database form that supports

common GWAS data management operations. The loading process has been implemented as an on-demand service to save the physical disk space. This approach enabled the researchers to select which GWAS dataset to be made available for the query operations. Keeping GWAS datasets "offline" (not in a database, but on-call for analyses) versus "online" (held in the columnar database for processing) approach has been adopted to manage the computing resources efficiently. The motivation is that GWAS datasets do not need to be in database form all the time; only when interactive data processing operations, such as querying, are required. To have the GWAS datasets in database form all the time would be inefficient in regard to disk storage, and would place a large and unnecessary load on the database system.

Query operations do not operate directly on the columnar database due to the binary data representation of the data within columns. The intermediary data transformation mechanism needs to convert query results to a selected GWAS data format and execute the complex query operations. For the SPARK interface to remain usable during this process, the transformation must happen transparently, as a background process, without forcing a waiting period on other operations (i.e. The Ark-specific subject creation, retrieving specific subject demographic information, generating a report, etc.). To meet these requirements, a software implementation based on the Lambda architecture was used [50]. This approach brings flexibility, as it adds an abstraction layer between the database and SPARK software layer and facilitates data operations that are defined as workflows independent of the database technology.

In addition to empirical benchmark results to support SPARK's data management approach, Section 7.5 provides a detailed comparison between this component of SPARK and competing systems. A SPARK-specific data compression mechanism is introduced which is unique to the implementation. A predefined query mechanism to extract the GWAS information, and online/offline data storing modes to maximum utilisation of computing hardware resources in GWAS data management

are implemented.

1.3.4 Fostering biomedical research collaboration

The first wave of the GWAS has been recognised as important to share GWAS datasets and analytical methods implemented by the different research groups [46]. The existing analytical platforms (SeqWare, Galaxy) are based on a single HPC platform and have limited the accessibility to local researchers. There has been a number of facts which have contributed to the less collaborative nature of the GWAS. The existing security constraints bound to the HPC facilities result in restricted accessibility for the wider research community. The privacy practices in the existing GWAS datasets limits the exposure of much of the collaborative research. The high coupling of the technical implementation of the analytical methods towards computing sources has required a new version to be implemented when importing to new execution platform. These computer-related activities require a specialised skill set.

The SPARK project has been designed to enhance the collaborative GWAS by introducing a study-based secure research data environment for The Ark (see Chapter 3). GWAS researchers can maintain a genomics study space to share with the collaborators and specify different access levels to share their datasets. Also, a simplified web user interface via The Ark genomic module has simplified the complex analytical pipelines. SPARK's computational package model provides a standard format for researchers to construct a self-contained implementation of an analysis algorithm. The installation of these packages links them to a particular study. This means that a researcher with access to a study and genomics module within The Ark can apply the algorithm to the study's GWAS datasets. In this way, SPARK serves researchers as a repository of genomic analysis packages to operate as an analytical and knowledge hub.

Genomic datasets are a key component of the analytical process. A number of

privacy and security issues arise when considering the problem of providing shared access to datasets, the methods to query and analyse them, and the results. The new genomics module provides the means for each study to connect it to SPARK nodes (via microservices); the configuration of one study is independent of other studies, and only a researcher with administration privileges has access to this configuration. For researchers, SPARK has simplified the data management of the GWAS by providing a virtual representation of the datasets, residing in data stores, via web interfaces (see Section 5.4.2).

Researchers can navigate the contents of shared data centres and shared datasets which are linked to particular studies and not visible to other studies. The approach of providing a unified, web-based view of decentralised data centres means that datasets can remain in their host data repositories, and only be transferred when querying, processing or analysis is to be performed. To perform an analysis, a researcher simply selects a dataset visible to the study that has been brought into context, selects an analysis package that has been registered with SPARK for the study, and initiates the analysis with a click of a button within the web-based interface. The results of the completed GWAS are then visible and shared to all researchers with access to the study and The Ark's genome module. Also, communication between The Ark, SPARK nodes, data centres and analysis platforms is encrypted using state-of-the-art technology, and therefore the information cannot be intercepted and decoded.

1.4 Thesis outline

The following chapters of this thesis will discuss the details of the research findings and the research process employed to produce SPARK, the major contribution of this project.

Chapter 2 provides a literature review, serving as the context and background upon which SPARK research is based. In this way, the development of SPARK

has been driven by a review of the state-of-the-art in similar systems, which encompasses journal publications, conference proceedings, books, and the supporting reference material of existing biomedical and analytical platforms.

A detailed discussion of The Ark is given in Chapter 3, discussing the common functionality of biomedical data management systems and how this relates to the capabilities of The Ark. The chapter also compares The Ark to other prominent biomedical data management systems (see Section 3.13), highlighting its merit and suitability as the basis for SPARK.

Chapter 4 examines the problem of pedigree data modelling and visualisation and contributes a new open-source pedigree module for The Ark. The chapter presents an efficient and generalised data model to capture pedigree structures (see Section 4.5), in addition to a new algorithm to infer family structures using the data model (see Section 4.6). Practical functionality, such as the ability to import/export and visualise family datasets are also covered (see Section 4.8).

Chapter 5 contributes a new architecture for a genomic analysis platform, i.e. the SPARK architecture. This architecture is a key component and substantial contribution of this thesis; underpinning SPARK as a study-based GWAS data management and knowledge discovery system that is integrated with The Ark. The architecture proposed allows installation of The Ark to interface with multiple SPARK nodes, where each SPARK node itself can interface multiple data centres and HPC platforms. Microservices are used to declare the independent service nodes specifically suited for the HPC platform requirements and provide an independent operational environment which will maximise the service availability.

The special challenges of Big Data management, in this case, GWAS data management, are addressed in Chapter 6. Section 6.6 demonstrates that a conventional approach using relational databases for GWAS data management is impractical. Section 6.7 discusses the types of data models that have been developed for Big Data management and provides the rationale for selecting Cassandra, a non-relational columnar database for SPARK. The results of benchmarking experiments are given

in Section 6.8, showing that the Cassandra database approach has reduced the GWAS data import time to a realistic time frame and saved much of the physical disk space by Cassandra columnar data model-based binary data representation. Section 6.9 presents a Lambda approach [50] to integrate the columnar data model with the microservice-based architecture of SPARK nodes. The use of the Lambda data architecture has provided an interpretation layer between the application and database tiers by enabling more efficient data representation and the information based on pre-configured query parameters to be extracted.

Chapter 7 compares and evaluates SPARK concerning similar systems, focusing on genome browsers (Ensemble browser, UCSC browser and dbGAP database) and analytical platforms like SeqWare and Galaxy which are specialised in GWAS analysis. The chapter discusses the capabilities and features of current SPARK implementation, illustrating the merit of SPARK's architectural design, a novelty in the use of a columnar model tailored GWAS data processing and integration with The Ark as a comprehensive biomedical data management solution. Sections 7.2, 7.3, and 7.5 demonstrate that while competing systems such as genome browsers and analytical platforms sometimes have more advanced features than SPARK (e.g. genomic data querying), the contribution of this thesis represents the first genomic data management and analysis platform that is integrated with a sophisticated study-based data management system, capable of working with not only GWAS data, but also non-genomic data such as phenotypes, consent and contact tracking data, biospecimen data, pedigree data, etc.

Finally, Chapter 8 provides concluding remarks for the thesis. Avenues of future work, including transforming the family relationships and family demographic information for GWAS, developing SPARK-specific GWAS analytical repository, and multi-study-based workflow management, are proposed in Section 8.2.

In summary, the SPARK project has introduced a novel web-based software system to accelerate advances in medical research aspects by implementing a knowledge discovery platform that enables HPC and Big Data management for hetero-

geneous medical datasets.

Systematic literature review to support the evolution of data management systems, informatics applied for the medical research data management, advances in medical research to Genome-wide Association Studies, and High-Performance Computing availability and application of research.

2

Literature Review

2.1 Introduction

This chapter reviews state-of-the-art approaches to biomedical data management and analysis, extending to the genome-wide association study (GWAS) scale. These approaches use novel software design to accommodate high-performance computing (HPC) and big data management for existing biomedical data management systems. GWAS data management and analysis within a study-based data management environment enables collaborative research through sharing existing data sets and analysis techniques facilitated by user-friendly access to expensive HPC resources via a general web user interface.

Section 2.2 examines data management as a general research concept, presenting

its historical progression to the present time. This is followed by a discussion of landmark events in the development of data management techniques and supporting data management systems. The section then discusses changes in dataset properties and the evolution of big data. The section concludes with a discussion of how data management systems handle large heterogeneous datasets, and in particular genomic datasets.

Section 2.3 discusses the latest systems used for informatics in the biomedical domain. Biomedical data management systems have been developed to manage specialised medical research datasets. These have rapidly evolved after adapting web technologies to satisfy the modern medical research requirements. Much of the focus of biomedical data management systems has given to phenotypic data management (e.g. demographics, questionnaire and biospecimens), while much less attention has been given to pedigree and genomic data management. While standalone systems have been developed to manage pedigree and genomic datasets, there is a need for software architectural design to integrate study-based biomedical data management systems with family and genomic data management.

Methodology for GWAS data management is presented in Section 2.4. Completion of the Human Genome Sequencing project has enabled the identification of susceptibility genes for a myriad of diseases. Previously, the major hurdle for conducting such analyses was the cost of genome sequencing technologies, but the introduction of affordable specialised single-nucleotide polymorphism (SNP) chips by commercial vendors has changed the genomic research landscape. GWAS findings are being used to inform personalised medicine that caters for individual patients' level. This approach is gaining popularity in the medical research community. To prove more significant GWAS results have required novel statistical methods and analytical approaches powered by the HPC sources.

The connection to HPC technologies is discussed in Section 2.5. HPC is a much-required resource for the complex data analysis, especially genomic data processing. Today, there are multiple HPC resources available to researchers, but these require

substantial learning and hands-on experience to work with them efficiently. There needs to be a systematic approach to reducing the overhead of interacting with HPC resources for genomic researchers who are not computer experts. By creating a collaborative resource pool, genomic researchers can be supported globally.

2.2 Data Management

This section presents the historical evolution of data management, the introduction of database management systems, more complex software systems and their role in managing research data, and the importance of biomedical informatics for medical research. Challenges in biomedical data management and limitations of existing database technologies are also discussed.

2.2.1 Data management systems

Until the end of the 19$^{\text{th}}$ century, data was stored on popular writing media including clay tablet, papyrus, parchment and paper [51]. These records were manually maintained by individuals and were fragile and easily damaged by the elements. The management and processing of the old data vaults required massive human resources in the absence of automation. At the beginning of the 20$^{\text{th}}$ century, machines were introduced to aid data processing, but paper remained the main storage medium [51]. In 1896 IBM was the first to introduce punch card machines to process data automatically via the use of binary patterns [52]. This approach was used until the mid-1950s and resulted in huge piles of punch card being stored in data processing centres. In addition to storage issues, the card handling process and massive manual processing time are taken to conduct data processing were problematic.

Enabled by revolutionised electronics in the 1950s, scientists developed magnetic tapes that could digitally store relatively large information sets [51]. These

tapes were an ideal replacement for punch card machines because they were capable of storing more than 10000 punch cards per tape. Supporting software systems were built to process the data held on magnetic tapes, and specialised computer languages were developed to handle data processing operations. These software systems were based on a file-oriented data model and were capable of handling batch transaction processing on periodic schedules. These systems were capable of representing the organisational file repository in a digital format. The data is organised using the metaphor of files arranged into hierarchies of directories, now often called folders in modern computing. A file is one unit of semantically meaningful information that can be used as input to a computer program. This approach was simple, transparent and well suited for simple organisations. But when this model was adopted for large business conglomerates and complex business scenarios, significant problems arose. These problems arise because the file-oriented model is incapable of avoiding data redundancy, increases the possibility of data inconsistency, has data that is closely coupled with access programs, has less data interoperability, has inadequate data sharing capability and has data security issues [53]. The inability to adapt implicit validation mechanisms to check for data duplication inside a file or set of files has resulted in data redundancy. Different file structures have created inconsistent data patterns in the file systems on sharing the information and created multiple data security issues due to the enabling file access to a large set of machines.

The success of magnetic tape data storage and significant improvements in computer hardware led to further advances in data storage systems. These systems enabled real-time information storage and retrieval based on the fundamental concept of a data model. Early data storage and retrieval were based on the hierarchical data model. These datasets were organised into individual tables with a single parental entity related to multiple child entities. These can be graphically represented as a parent-child rooted tree structure. Hierarchical data models suffer from data redundancies and update issues in data management systems. The hierarchi-

cal data model causes some data elements to be repeated in the different datasets, and to maintain data consistency every dataset must be updated when a repeated element changes. An alternative to the hierarchical data model is a network model, in which data entities are represented as a connected network graph mapped to the logical computing process. There can be multiple parent entities per child entity, unlike the single parental entity limitation of the original hierarchical model. In parallel to the development of the network model, new programming languages introduced the concept of schemas to separate the logical and physical datasets. Data schemas represent the blueprint of the data element placements inside the database and describe the logical relationships among the stored datasets [54]. While the network model received a lot of community support [51], some researchers argued that the model is too tightly coupled with the software interface layer. That is, every time there is a change at the software application layer (i.e. the database client), the table schema of the network database must be modified to accommodate the change. This limitation led to the introduction of the relational model for database management systems.

The relational model for data management systems places emphasis on the independence of the data from the application software level [15]. The model defines tables as interrelated sets, and relationships are defined as the subset of intersection in the table sets. A relational model presents n tuples of relationships in the table rows [15]. Similar to the network model, the relational model adopts a table structure to store the datasets. Each table consists of multiple rows to store the data elements. Table columns represent the individual properties of the data rows. Relationships between rows in different tables are defined by linking primary keys established in table columns. The relational model has addressed the contemporary data management problems of ordering dependency, indexing dependency and access path dependency [15]. Ordering dependency is created when data models are forced to store records according to predefined order parameter. Indexing dependency is created when every data element has to be indexed in the tables,

and every time user makes a change to the data the index has to be updated. Data dependency causes significant performance bottlenecks when dealing with the large datasets. Access path dependencies were introduced when data models were tightly coupled with the program layer, and the table column order has to be exactly matched with the data request. In the early 1970s, these issues were common with data management systems.

Codd has proposed declaring the relationships by adding columns to the tables [15]. One additional column defined in a table to maintain a relationship is called a unary relationship. Similarly, adding more columns to maintain the relationships are called binary, ternary, etc. Complex data domains require a normalised data form to avoid repeating data elements in different tables. In a normalised form, data is organised into individual tables and relationships are represented by adding foreign keys. Figure 2.1 presents a general picture of a normalised dataset. Here Company and Users are selected entities, and one or many users can belong to a company. The relationship between a user and a company is represented by the Company ID element in the User entity. This relationship is externalised by the normalised relational data model, and by selecting a user or company, entities can check which company the selected user belongs to or which set of users belong to the selected company. The User and Company entities can insert, update and delete individually without affecting the entity relationships.

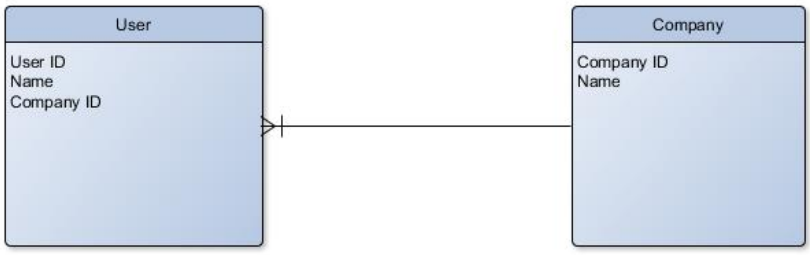


Figure 2.1: A normalised representation of a dataset.

Codd proposed a universal data language to extract information stored in relational database management systems [15]. The language, named SEQUEL, was

introduced by IBM in 1974 [55], and has been used to change and extract data from IBM databases. Based on relational algebra, and following the development of SEQUEL, the American National Standardized Institute (ANSI) proposed the Structured Query Language (SQL) specification to be applicable to all relational databases [55]. The SQL specification included definitions for primitive data types, a data definition language to create database objects, a data manipulation language to maintain the data defined in a database and a data control language to control user access privileges. In general terms, SQL facilitated the development of relational database implementations that allowed creation of the data holding objects; enabled data to be created, retrieved, updated and deleted; and were capable of maintaining data security. The SQL ANSI standard has been amended twice-in 1992 and 1999 – to satisfy the original goals of SQL and extend its capabilities to ever-growing database requirements. Many modern relational database systems follow the ANSI SQL standard, which allows database developers to work with a common programming paradigm and has played a significant role in the popularity of relational database management systems.

The popularity of websites backed by databases has increased the operational workload for relational database management systems [56]. Due to these new demands, relational systems have required important new capabilities, including transaction control and multi-user environments. A database transaction can be defined as the total database operations needed to be completed for a task. Database transactions are crucial to maintaining data integrity in large databases and enable synchronisation with multiple clients. A transaction control system will only commit database changes if data integrity is satisfied for the entire transaction. Otherwise, it will roll back partial changes and return the database to its state before the transaction. Simultaneously connecting multiple users to a database creates a multi-user database environment that needs to satisfy data concurrency issues, deadlock avoidance, security measures and transaction handling to avoid unexpected results.

Database-backed websites with multi-user login capabilities depend heavily on simultaneous user login actions, individual data privacy and restriction to direct database access by end users. These actions create a large number of database requests that the systems need to be able to support. The relevance of these features will be discussed in Section 2.3.2, in which web-based systems for biomedical management are reviewed.

Object-oriented programming languages gained popularity in the 1990s, in part due to the rapid expansion of the world wide web. Object-oriented languages are based on five design principles called SOLID [57], which provide freedom to developers in independent implementation, higher code reusability and less maintenance cost compared with procedural languages. SOLID's five basic principles are: (1) single responsibility – the object-oriented class/module structure should be responsible for only one functional aspect of software specification; (2) open/close – the class structures should always be open for extension by inheritance but less prone for the modifications; (3) Liskov substitution –the classes are represented by object instances in the execution environment, which can be dynamically over-ridden by the sub-instances without altering the design goals; (4) interface segregation – creating multiple custom interfaces is better than keeping a single generic interface; and (5) dependency inversion – the importance of decoupling software modules by abstracting the design without detailing the individual component implementation. As computer systems became more powerful, they enabled various e-research experiments in different domains including e-commerce and medical research. This type of work involved complex and large data types (such as large text files, image data sets, streaming data and experiment outcome data) to be stored in databases that were typically relational. Most of the data generated from these domains are unstructured and exceptionally large. Unstructured datasets consist of unknown data patterns, repeating data elements across the dataset and data formats that are not feasible for the query engines. The relational data model has encountered significant performance and storage issues when trying to store unstructured datasets. To

address these challenges, database researchers proposed new multimedia database designs including object-relational databases, extensible markup language (XML) databases and other domain-specific database types [58].

Object-relational databases were introduced to support object-oriented programming. Object-relational databases were specially designed to store the current object status on the object orient principles. They store the available object instances and later retrieve them directly to the application context. Another advantage of these databases is that they avoid the impedance mismatch: the data type mismatch between databases and programming languages. The implementation of object-relational databases was undertaken using two approaches: the first approach was to build a pure object-relational model, while the second approach used an object model mixed with existing relational databases [59]. Both approaches saw a degree of success, and several databases such as Illustra and Omniscience were introduced to the market [60]. However, the object-relational approach was not widely accepted by the database community due to the development of languages that eliminated the impedance mismatch problem. Furthermore, the object-relational approach reduced the degree of data independence from application programming interfaces and was often difficult to fit existing relational databases and datasets [59].

Object-relational databases have lost most of their market share due to the changes made to the SQL specification in 1999 [58]. Since the late 1990s, XML has become the most widely accepted data interchange language, gaining widespread use in e-business applications [58]. XML data is presented in a semi-structured format and has become the de facto communication language for web services. While specialised databases have been developed to store XML documents, some vendors of relational database systems have introduced a native XML data type and a shredding' mechanism to decompose and store XML data in existing relational environments [58].

Irrespective of the data model used, the number and size of datasets are ever-

increasing, thus demanding new database technologies that will scale accordingly. For example, popular web applications such as web crawlers and analytical engines generate a massive amount of data per second that cannot be stored on a single server [34]. Recent advances in computational power have enabled science researchers to explore new domains and generate datasets that are larger than the capacity of standalone storage disks. To address these challenges, large distributed databases backed by large amounts of computational power have been developed. These distributed databases are designed to store data across storage clusters that may span multiple locations. The design of these databases ensures that data retrieval and processing is not affected by the distributed nature of the database architecture.

The development of distributed databases is based upon the concepts of wide applicability, high scalability, high-performance and high availability [61]. The relational data model was shown to suffer performance bottlenecks when implemented in a distributed database architecture, leading to speculation that a new data model is required for these databases [61]. Therefore, unique data models tailored to distributed architectures were developed to support distributed databases. Data models that are based on a key-value pair, column oriented and document based are quite popular in the distributed database context [62]. Their unstructured data handling capability and distributed architecture avoids using SQL to handle the data stored in these databases. Therefore, these data models consist of a unique programming interface to handle the data residing in distributed databases and are commonly known as NoSQL databases. Google's BigTable is a popular example of a distributed database that supports a large column family based data model [61]. BigTable provides a unique advanced programming interface (API) to access data elements and manage the database functionalities [61]. Section 2.2.3 discusses NoSQL databases in more detail.

The emerging popularity of distributed database systems has led to a substantial change in database hosting approaches. A traditional data and application

deployment has the database hosted alongside the application server on the same local hardware. Hosting the application and the database on the same hardware requires significant resources for system monitoring and maintenance. In addition, there are security, reliability and load balancing issues caused by the localised infrastructure (e.g. being unable to switch servers when there is a malfunction). As an alternative, Cloud-based data services have gained popularity in recent years due to the ease of maintenance, cost effectiveness and the often large amount of computational power required [63]. Cloud-based solutions enable small-scale application vendors to host data management systems on a large-scale managed hardware infrastructure. Database and application server stability and reliability have always depended on the stability of the internal infrastructure, and with cloud-based solutions, stability is managed by dedicated support consultants rather than localised support which may be at times *ad hoc*. While some have expressed concern over private data being hosted on third party cloud platforms, specialised system design techniques and legal agreements between the cloud providers and clients may be implemented to address this issue.

Cloud resources typically provide a tailored API for applications and programmers to communicate with the services; this concept is popular among application service providing companies such as Google, Amazon and Microsoft [64]. For example, developers may implement custom data extraction methods to access existing datasets. Technology analysts predict that cloud-based platforms and data management APIs will be a major database technology of the future due to their cost effectiveness, storage capabilities, available power and track record of success [64].

The rapid evolution of computer technology and its use in scientific research has created a large number of multi-dimensional datasets that may be very large. Such datasets have four common attributes [65]:

Volume — size of the dataset

The sheer volume of generated data has resulted in massive data sets. Few reasons behind the generations of such datasets include; the number of years

of existence, association with specialized sources, and the domain specific requirements. The recent technical advances in the internet era has significantly contributed to certain dataset measurements (internet search patterns, current weather details, email servers, etc) which scales up to the petabytes.

Velocity — frequency of the dataset

The time taken to update a dataset contributes to defining the velocity of the dataset. In modern day applications, datasets are updated frequently due to widespread internet connections and large user distribution across the internet. Some applications are updated more frequently compared to regular applications. Common examples are business transaction management systems (e.g. Ebay[1]), email clients (e.g. Gmail[2]), and weather reporting systems (e.g. weather.com[3]).

Variety — complexity of the dataset

Specialised software systems have been developed to collect information on the internet content and user behaviour. Web crawlers are a typical example where information is gathered from published content and datasets are generated without checking the preconditions of data formats. Google web crawlers are quite popular and generate complex data formats including character streams, image streams and video streams. There is no data unification or categorisation mechanism implemented in these datasets in the relational data models due to no predefined data templates for the external web resources.

Veracity — legitimacy of the dataset

The legitimacy of existing datasets played a pivotal role in ensuring the accuracy of the decisions made from them. Large datasets generated from the applications like web crawlers have significant decision constraints because

[1] `www.ebay.com`
[2] `www.gmail.com`
[3] `www.weather.com`

of their unknown data templates and content. Therefore, intelligent data validation mechanisms are needed to justify the accuracy of the results.

Any dataset that has one or more of these characteristics may be referred to as 'big data'. Despite the increases in computational power and storage, relational databases cannot sufficiently model big data. Section 2.2.2 discusses the limits of relational database systems, the challenges of organising non-structured datasets into a relational table model and an alternative approach to handling big data inside non-relational databases.

2.2.2 Relational database limitations

The introduction of the world wide web and rapid developments in computer hardware have produced large datasets in the business and medical research contexts. In the medical domain, these developments have seen the introduction of full sequence and chip-based genomic datasets. Extracting meaningful information from these large heterogeneous datasets has become a significant ongoing challenge [51]. When using a relational database model, it is necessary to understand the context and structure of the data before being able to store the data in a database. The time taken to gain an understanding of the data context and structure has a significant impact on data management costs. The major challenges for the relational database schema are to scale up to the big data set size, timeliness of data extraction and efficient management of disk space.

Datasets generated in a data-driven environment such as medical research often feature unclear data schemas, need rapid changes to existing schemas and can have unconventional schema structures [66]. Relational schemas are created based upon clearly structured datasets with an expectation of very few schema changes over the lifetime of the data. Modern datasets can demand unorthodox changes to a schema structure, and this demand poses a significant challenge to the continued use of the relational data model for big data. Future dataset guarantees cannot be

made on precise data definitions and reliability of data structures [66]. Relational databases also employ optimisation techniques to filter the information stored and perform indexing for rapid querying and manipulation. These optimisations cannot be applied to the unstructured and dynamic data sets generated by modern day applications, leading to significant performance tuning problems for such datasets managed within a relational environment. These datasets have been stored as character large objects or byte large objects in relational databases, and these data types (discussed in more detail below) are not bound to the optimisation techniques.

As discussed in Section 2.2.1, distributed database systems are a means to distribute the load of database management for large-scale datasets over a number of servers or server clusters. These customised distributed data models facilitate the storage of large datasets and provide a means for the increased power of massively parallel computational resources to process the data [61]. The server cluster provides a thin layer of application execution environment that shows as a single computer execution environment to the application but physically consists of several computing processors and disk spaces. Relational database management systems were originally designed to operate on a single server infrastructure [62], and this means that the relational model faces significant performance and architectural issues when applied in a distributed architecture [63]. Applying the relational model in a distributed environment includes difficult make table joins.

A significant drawback of the relational model is that it requires significant processing time to execute complex queries on a big data scale dataset [67]. As an example, searching a web crawler data repository in a relational environment, which usually has petabytes of records, would take a significant amount of time to complete the query [61]. This scenario is quite similar for GWAS datasets, which have data for millions of SNPs from large population samples. A sample GWAS relational data schema is presented in Figure 2.2. This problem is made more severe due to the sheer size and complexity of modern big data, which in turn demands highly complex, and often highly nested queries, to extract meaningful information.

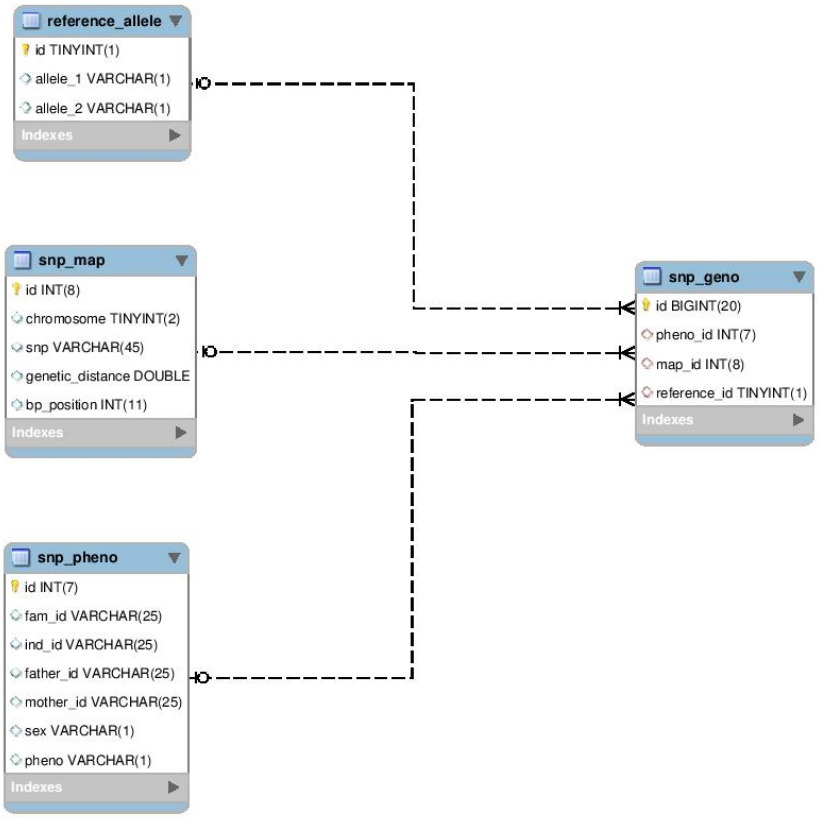


Figure 2.2: An example relational database schema to store GWAS data.

The relational model also requires a processing mechanism to handle non-structured datatypes (i.e. fields that can in hold information internally). For example, standalone SQL cannot easily extract information stored in a binary object (character large object or byte large object format because this information is simply seen as a sequence of data bytes, without any interpretation or meaning placed on the data contained in the sequence. A second level of matching algorithm is therefore needed to combine structured results with the information stored in non-structured (from the SQL point of view) components, and this requires additional programming effort [67]. Additional programming poses a non-trivial challenge to developers and may demand significant architectural changes to the application during the software development process. Moreover, because SQL is the only means for information extraction in a relational database environment, an

increase in data complexity and relational schemas leads directly to an increase in more complex SQL code, which in turn affects the modern agile software development life cycle [62].

In summary, the limitations of the relational data model can be categorised into four groups [62]:

Scaling : The relational model faces operational issues in distributed computing infrastructures and cannot readily use distributed computational power.

Complexity : The problem of storing and querying non-structured data in a relational environment.

SQL : The limitations of SQL when operating with non-structured data such as binary objects (e.g. genome sequences, images and text passages).

Features : Relational data environments often have a large amount of rarely used and unnecessary features, especially for non-structured data. These features may impose a performance cost to the database.

These issues demonstrate that, while the relational model caters well to some data, it is not suitable for storing large non-structured datasets. Consequently, customised data models need to be used to store, manipulate and query this data efficiently as well as make use of distributed storage and computational architectures.

2.2.3 Current approaches to manage high-dimensional datasets

In genomics, rapid advancements in computational power and storage and large cost reductions in sequencing has to lead to demand for platforms to handle massive high-dimensional genomic data [68]. Because the size of each dataset is often very large, with multiple attributes per data unit, the data requires specialised

management techniques to facilitate the extraction of meaningful information. As discussed in Section 2.2.2, today's relational databases cannot face this challenge due to the massive size of the datasets and ill fit to the relational model. Therefore, individual domain contexts have introduced new database models and techniques to handle high-dimensional datasets. Examples of these database models are now discussed.

Google BigTable

Google, a web technology company, has extensive experience in managing extremely large volumes of high-dimensional data [61]. Google receives petabytes of data from propriety applications including web crawlers, search indexers, Javascript-based analytics engines, satellite based geospatial engines. These data do not fit in a single server relational database because single server databases are currently limited to terabytes at most. Furthermore, even it is possible to store the data, the computational power and memory limitations of a single server would be insufficient to process or query the data. Consequently, Google developed a distributed database management system dubbed BigTable. BigTable provides a cluster-based HPC mechanism to power data management, while the data model consists of a row, column and timestamp three-dimensional architecture. This implementation includes a custom access API that supports the MapReduce parallel programming model, also developed by Google. The MapReduce model enables the process to be sped up by parallel processing the individual datasets. The BigTable implementation uses the Google file system [69] as the main storage medium, facilitating automatic compression and schema management within the system. BigTable is a proprietary implementation developed by the Google that has the MapReduce model embedded. The open-source version of the MapReduce model (known as the Hadoop framework) was implemented by the Apache foundation and supports the HBase big data management system.

To function as a highly reliable and highly available data management system for large structured datasets, BigTable uses column family-based locality by grouping similar data type columns together and separately maintains keys to these column groups. A data compression mechanism increases storage efficiency, and a data-caching mechanism improves information availability. Bloom filter techniques are used to improve read operations. Bloom filter is an efficient probabilistic data structure that is used to check if an element resides in a set. The Bloom filter returns only the false positive results for selected set. BigTable uses an individual commit log for each table, along with a high-speed table recovery facility for fault recovery. The database employs a metadata-based storage mechanism to improve table immutability. This means that any update made to the data will create a new entry in the database and the metadata will be updated regarding the new data location. Metadata reduces the time taken to complete a database transaction and increases the data availability to the users.

The BigTable model has been highly successful in Google Analytics, Google Finance and many other large data products built by Google [61]. The success of BigTable in real world applications highlights the importance of developing unique data management technologies. Google's implementation shows the importance of considering distributed data management failover facilities and adopting software best practices when developing heterogeneous data management systems. The relatively simple architecture of BigTable positions it well to face future enhancement and extension.

SeqWare Query Engine

The SeqWare Query Engine was developed using big data technologies to store and process raw genomic datasets in a distributed computing environment [70]. SeqWare uses a Hadoop-based [71] parallel computing facility to store large genomic datasets in an HBase [72] multi-column database. Hadoop is the open-source im-

plementation of Google MapReduce framework and has provided the facility for distributed storage for massively large databases. HBase is an alternative multi-column database implementation to that is suitable for large datasets with a large number of columns. SNP datasets usually consist of hundreds of thousands of base pairs, and HBase is able to handle these types of datasets.

Using parallel computing power at the database level brings immediate access to the massive processing capability that large heterogeneous genomic datasets typically demand. This approach is substantially more effective than adding more and more memory and computing resources to a relational database that inherently doesn't scale well and cannot fit a distributed architecture. SeqWare has made performance comparisons with distributed and non-distributed database back ends and shown a significant time advantage when querying large datasets in a distributed environment [70].

According to the SeqWare documentation[4], it provides open-source based automated workflow management at the hardware level rather than a more user-friendly interactive web front end. The SPARK project will directly address this limitation, and the SPARK implementation architecture will be benchmarked with the SeqWare system (see chapter 7).

NASA Earth Observing System Data and Information System

NASA is another example of an organisation that must manage and process extremely large volumes of data. The NASA Earth Observing System Data and Information System (EOSDIS) manages data obtained from a variety of resources that monitor global climate changes [73]. This project primarily gathers information from satellite observations transmitted to ground information centres. These datasets reach petabytes in size and are managed by thirteen data centres across the United States of America.

[4] `http://seqware.github.io`

The objective of EOSDIS is to implement a centrally managed database system to handle distributed datasets across global data centres. EOSDIS provides a web interface for earth science researchers to access large heterogeneous datasets. The EOSDIS architecture achieves this by introducing service-based data locator facilities and providing a territory data storage mechanism. EOSDIS facilitates customised data schemas that are maintained at different data warehouses and updated by the central node using the unique features of each schema. Each territory's storage facility is capable of data compression, thus increasing storage efficiency, and is responsible for uploading each site's metadata to the central server.

EOSDIS allows researchers to create customised queries via the web interface. The web interface provides details about the available data, using unified representations of metadata held at the central server. Using a standard login process, researchers can create custom queries based on metadata summaries, and this has enabled researchers to track their jobs and reuse the custom queries. This has created a global earth science data platform for distinct researchers by creating centralised heterogeneous data repository consist of user based access privileges, common knowledge sharing capabilities and data processing mechanisms (cleaning, analysing and filtering).

NCBI Database of Genotypes and Phenotypes

Returning to the genomic domain, the large size of unstructured genomic data matched with structured phenotypic information requires a mixed model data management system. To address this requirement, National Center for Informatics developed the Database of Genotypes and Phenotypes (dbGAP) repository to manage phenotypic and genotypic data using a relational data environment [74, 42]. dbGAP provides researchers with a means to store phenotypic information in relational table structures and related genotypic information as compressed binary objects in

relational tables.

dbGAP operated as a public data warehouse for genomic research studies. Also provides private workspace for the research data by providing the role based access level security restrictions for the individual dbGAP users. The datasets in the system have been grouped into study and related sub-studies by specifying individual security access levels. The general public can search and extract results from the dbGAP public data schemas, but only authorised users can access the private data repositories. The system consists of a search mechanism based on Boolean operators, and researchers can extract information based on existing and custom queries.

dbGAP is based on the relational data model and has to keep genomic data as compressed byte large object archives. dbGAP requires custom scripts to decompress existing datasets and another set of scripts is needed to perform data analysis. Datasets submitted to dbGAP are analysed and cleaned before they are published to the system. Using a separate script limits the full potential of relational databases for the genomic datasets and introduces more work for the researchers. A non-relational data storing mechanism suitable for unstructured genomic datasets (potentially a NoSQL-based large database model) will eliminate the extra overheads for storing, processing and querying operations.

University of California, Santa Cruz Genome Browser

The University of California, Santa Cruz (UCSC) Genome Browser[5] supports searching and analysing genomic datasets. Currently, the UCSC Genome Browser consists of genomic datasets for 93 different species [39]. Researchers can carry out their analysis based on species and chromosome positions with the associated datasets available in UCSC Genome Browser. The datasets available in the UCSC Genome

[5] `http://genome.ucsc.edu`

Browser are hosted in UCSC servers under the UCSC administration. Researchers have the capability to save the current status of their study and resume the work from the previous saved status.

The UCSC Genome Browser supports third-party analysis libraries including BLAT; a sequence alignment similarity checking tools to carry out search activities on selected datasets. The UCSC Genome Browser search engine consists of a web portal to interactively carry out analysis activities (including visualising the results) and many command line tools to support the specific data activities (including data conversion and quality control). The UCSC Genome Browser has a stand-alone version, which can be downloaded and installed onto a desktop computer or server computer. The data import facilities support popular genomic data file types including PLINK binary format and Wiggle track format.

Ensembl genome browser

The Ensembl genome browser [38] stores genomic information on 87 species and provides search options to traverse the genomic regions by chromosomes and base pair positions. Ensembl is the only open-source genome browser available for researchers and it has been integrated with a number of open-source genome analysis tools. Datasets and analytical tools are updated periodically, and improvements are published via the Ensembl web portal (`ww.ensembl.org`). Popular tools integrated with the Ensembl browser include Varian Effect Predictor, BioMart data mining tool to analyse the annotations and datasets, BLAST genomics search engine [75] and the REST base programming API to allow third party tools to interact with the Ensembl browser. Enabling a programming interface has allowed researchers to develop analytical methods and run these in the Ensembl infrastructure. Data import and export facilities enable data owners to import their datasets to Ensembl and run analyses via the Ensembl cloud. Researchers can store their queries and

export the result data as text files with known genomic formats or results as an image diagram.

As a common practice, a mixture of relational and non-relational data management systems are utilized to manage the high-dimensional datasets. The relational data model has limited accommodation for high-dimensional datasets by providing only the character large object or byte large object. Security is a key constraint in managing specialised high-dimensional datasets in genomic research. Having enough computational power to enable analysis of a high-dimensional dataset within an acceptable time frame is another challenge. The systems that have been developed to manage and analyse high-dimensional datasets are equipped with mechanisms to handle big data sets and use the high-performance computing power required for the analysis in a secure data management environment. The systems discussed in the literature review are pre-configured hardware systems that consist of a large number of static web pages developed to interface predefined set of functionalities. The SPARK design is more dynamic with multiple users (potentially dozens per installation) (See Chapter 5) and numerous datasets that are brought online for querying and processing before being taken offline soon after (See Chapter 6). The relational approach would not work for SPARK, but it does work for dbGAP and UCSC Genome Browser because they are static. The next section will further discuss the biomedical data management practice in the medical research domain and existing approaches to managing the medical research datasets in modern data management systems. This will be followed by discussion of the software approaches needed to take biomedical datasets beyond the capabilities of traditional database management systems.

2.3 Informatics applied to medical research and practice

In 2004, a US Food and Drug Administration report stated that the effective medical innovation rate was slowing and highlighted the need for novel medical research tools and technologies [29]. The FDA explained that, although the human genome is now fully sequenced and large volumes of genomic data have been produced at high expense, much of the information contained in the genomic data has not been communicated or translated to the medical community. The authors argued that this is in large part due to the lack of tools to analyse the data produced by advanced sequencing techniques and proposed the development of a set of advanced informatics tools to manage large translational research datasets. Work to this end has progressed, and a 2008 Food and Drug Administration progress summary report [76] discussed several prominent bioinformatics initiatives made possible through collaborative partnerships.

In the era of advancing data management and software systems, researchers tend to investigate the opportunities for medical research and practice. Data is an important commodity in the medical research, and it has proven the importance in many cases. Biomedical informatics research area has emerged in order to observe the impact and the advancement in medical research. Furthermore, informatics tools and techniques were introduced to address the existing challenges in medical science.

Recent advancements in genomic research has increased researchers' interest in using advanced informatics techniques for medical research data management and analysis. A potential outcome of genomic research is personalised medical treatment. Personalised medical treatments have a critical demand for developing informatics tools to access and store the patient information. This required biomedical informatics research to find out and investigate the best possible use of existing BigData technologies, computational algorithms, and software architectural best

practices.

2.3.1 Biomedical Informatics

Biomedical informatics is the application of information science to solve biomedical problems. Bioinformatics research dates back to the 1960s when medical researchers first used computers to store, share and model their datasets [28]. Data was considered a primary research asset, leading to the application of information theory to extract meaningful information from existing datasets [77]. Today, the goal of biomedical informatics remains the same: to apply information science as a means to overcome current clinical medicine limitations and advance medical knowledge.

Biomedical informatics has been applied to various clinical diseases including cancer, diabetes and asthma. For example, information science has been applied in asthma research to implement new diagnosis, prevention, monitoring and decision support systems [78]. The extended capacity of genomic studies has resulted in dispersed knowledgebases that are centred about a variety of genomic regions. Identification of common genomic regions requires a common data platform to share the knowledge. Once such platform is caGrid, a system that aims to enable GWAS researchers to share datasets and compare research findings [79]. The next subsection is focused on systems designed for medical data management.

2.3.2 Informatics approaches to medical data management

State-of-the-art data management systems are essential to extract meaningful information from the typically large and highly structured biomedical datasets that are common in contemporary medical research. The key aspects of biomedical data management systems are data collaboration, security, reporting, validation, auditing, document management, storage, data backup and import/export facilities [31]. In addition to these design aspects, biomedical data management solutions typically

require unique features that are study specific. For example, The Ark (see Chapter 3) and REDCap systems are capable of handling multiple concurrent projects and diverse data collections by way of features that permit per-study customisations.

This section examines existing medical phenotypic and genotypic data management systems and discusses their approaches to managing medical datasets. The development goals of these medical data management systems were derived from researcher workflows to manage phenotypic and genotypic data.

The REDCap project[6]

Understanding the need to manage the biomedical studies with a number of medical research datasets have encouraged the REDCap system to be built [31]. The REDCap developments were started in 2004 by clinical researchers at Vanderbilt University, Tennessee United States. Based on clinical research data management experience, the system was built to support the following data management requirements.

A centralised data repository accessed by distributed researchers via the world wide web

A centralised data repository allows multiple study datasets to be hosted in a single database instance. Each study dataset consists of a set of individual data fields that are pre-configured inside the REDCap system. The study dataset is populated to the REDCap system after configuration to accommodate study-specific aspects of the data.

[6] `https://www.project-redcap.org`

User role based security and other security constraints declared to protect the hosted study datasets

Security is a key constraint in the medical data management domain. Therefore, authenticating user access to system resources and authorising user privileges to access study datasets are important aspects of biomedical data management. The REDCap system implements security infrastructure inside its design specifications and enables user roles to limit access to study data to authorised users.

Enabling the key data management functionality that is unique to biomedical datasets

Biomedical datasets have a unique set of characteristics that need to be considered design specifications. When managing a biomedical dataset using a software system, it is important to : (1) be able to import data from various sources , (2) maintain a proper track record of changes to the existing data elements via an audit trail, (3) implement data validation and quality control processes to ensure a reliable dataset, (4) include an attachment storage facility to keep the patient-related reference material such as consent forms, medical reports, questionnaires, mammograms, (5) make data backups in a secure environment to avoid data loss and (6) implement data output processes suitable for analysis in popular statistical packages (e.g. R, SPSS and Stata).

When a study has chosen REDCap as the preferred data management repository, initial steps would be creating electronic data capture forms for the study. The electronic data capture forms represent the study data field attributes with relevant validation mechanisms. The REDCap system has been designed to accommodate multiple sources of datasets per study. Therefore, multiple electronic data capture forms will be required to meet the research goals of the study. The REDCap data capture process is capable of using the electronic data capture forms in online or offline basis and event driven questionnaire model. The data captured via the electronic data capture forms can be queried using custom queries devel-

oped by the researchers and data can be exported in different formats suitable for third party analytical engines (e.g. Stata, R and SPSS).

The REDCap system satisfies the requirements for a centralised biomedical multiple study data repository that is secure and provides most of the biomedical data management functionality required by researchers. At the time of writing, 2785 research institutions in 119 countries are using the REDCap system. These research institutions host 527000 projects in the REDCap systems and 695000 researchers interact with the systems. According to the literature and web resources [80], the REDCap system is currently one of the leading biomedical data management systems from biomedical researchers.

REDCap does have notable limitations; in particular, the generic nature of the system's design does not capture the semantic meaning of data fields and dataset types. This can be problematic because much biomedical data has special semantic meaning, individually and between data fields. For example, REDCap does not provide pedigree data management by specifying the relationships between the individual subjects and visualising the pedigree relationships. REDCap does not have modules to manage highly specialised medical data such as biocollection inventory or biospecimen information that is typically managed by a laboratory information management system, nor can it handle genomic data. Finally, the REDCap project operates as closed source solution that is visible only for consortium members who have registered with the system administration.

The Ark[7]

The Ark [30] is an open-source web-based biomedical data management system that was initiated by Centre for Genetic Origins of Health and Disease at the University of Western Australia. The Ark project was undertaken to address the challenges of maintaining proprietary medical data management system. The Ark

[7] `http://sphinx.org.au/the-ark`

has been designed to address the semantic meaning of the medical data sets using a hybrid approach. This includes a generic data management approach for the common medical datasets (which is similar to the REDCap electronic data capture forms) and specialised data management approaches for the data sets requiring semantic meaning. for example, parental relationships that can be used to form family structures.

Chapter 3 of the thesis discusses the individual module architecture of The Ark and its specialised data management modules (e.g. study, subject, laboratory information, questionnaire and work tracking) to handle multiple data types with specialised functionality.

OpenClinica[8]

OpenClinica [81] is another web-based electronic data capture system that has been developed to support clinical research. Unlike REDCap and The Ark, this system is particularly focused on supporting clinical trials. OpenClinica is capable of declaring multiple studies and capturing multiple clinical trials per study. Clinical trial data is recorded based on the pre-configured set of electronic data capture forms. In additions, the system is capable of managing the study participant appointment bookings and can monitor the progress of individual trials. Data import and export capabilities are also included.

OpenClinica has a user role-based security architecture to manage the data security. System design follows the extension-based architecture to enable plug and play additions to the system. Pluggable components allow developers to add features that are compliant with the core functionality. The source code for the core modules is hosted in the public GitHub repository and managed by the OpenClinica community.

OpenClinica operates on a mixed licensing agreement. Core modules can be

[8] `https://www.openclinica.com`

hosted on local servers under an open-source license. Users need to purchase enterprise edition extensions to use the full features of the system. Clinicians can either purchase the extensions or make use of the core system functionalities with technical support from the OpenClinica developers. Another approach would use a local development team to develop the additional features on top of the OpenClinica core modules and maintain the system as an in-house development project.

Slim-Prim

Slim-Prim [82] is a biomedical data management system that supports a single study instance. In the single study instance, one installation of the Slim-Prim manages one study. Slim-Prim was developed by the Health Science Centre at the University of Tennessee. Based on the concept of a pre-configured data extraction tool from existing data sources, the system is designed to create data capture forms from well-known medical dictionaries (Ex: ICD-9/10). According to the pre-configured concept mapping queries, Slim-Prim can be used to extract results according to chosen formats (e.g. R, Stata and SPSS).

Currently, Slim-Prim operates on the Oracle 11g relational database server under closed-source licensing. Therefore, users are limited to University of Tennessee affiliates and parties who have registered with the university. Slim-Prim avoids accommodating medical datasets managed by international researcher groups due to the ethical and legal constraints. Slim-Prim has been designed as web-based tools to support data extraction tasks, and it provides the mechanism to upload concept maps as text files, Excel spreadsheets or pre-configured database schemas.

While Slim-Prim supports the query engine existing datasets, it does not address the requirement of understanding medical data semantics. The REDCap and the Ark systems have been developed to handle the medical data semantics by their unique database schema design and functional implementation of specific data management processes. Slim-Prim only provides a query engine based on

the existing medical dictionaries and operates as an intermediary data pipeline to third-party analysis tools. In the long run, Slim-Prim design let researchers interact with multiple data schemas per study datasets and compliant with diversified query specifications for each study dataset. This would lead to a computational overhead in data management and inability to manage the existing datasets in a centralised data management system. Also, de-centralised data schemas have increased the complexity and decrease the reusability of the data mining and analysis operations.

Translational Data Mart

Translational Data Mart (TraM) is a single study data management repository developed by the University of Chicago [83]. Similar to Slim-Prim, the TraM system operates as a semi-automated workflow management system based on preconfigured data sources. TraM supports workflow management of multiple biomedical datasets including study participants, study questionnaires, biospecimen and clinical trial data. In addition to workflow management, TraM uses user role security architecture to provide a secure environment for data and workflows. The project operates as a closed-source project and is available only for the registered users.

TraM is not suitable for managing real-time data sources, however semi-automated approach has been developed to manage heterogeneous medical data sets. As a single study management system, TraM can handle only one study data set per installation. As a closed-source project, TraM is only available to University of Chicago researchers. The closed-source system developments and institution-based installation will introduce additional ethical and security concerns when migrating the external medical datasets to The TraM.

Section 3.13 presents a comprehensive comparison of the features of the biomedical data management systems discussed above. The comparison serves to highlight

the features required for a successful biomedical data management system and the need for a system that integrates analysis facilities with the management of typical participant data (e.g. consent status, contact information and questionnaire responses), pedigree structures and genomic data. Furthermore, those tools and systems focused on the systematic approach to managing the existing phenotypic datasets which integrated into a single relational database management system. Genomic data are semantically different from the phenotypic datasets in terms of their size and complexity. Section 2.2.3 discusses genomic databases and analytical platforms that contribute to management in of the high-dimensional datasets. These include the UCSC Genome Browser and Ensemble genome browser for managing full genome sequence data of living organisms, the dbGAP data warehouse develop to manage the genotypic datasets in the public web space and the SeqWare analytical engine that was developed to manage and execute genomic analyses. The common features for these systems are: (1) a hybrid approach to managing genotype datasets with a combination of relational and non-relational databases, (2) integration with HPC resources to execute the analysis and (3) a common application programming interface to develop custom genomic research tools. In addition to the systems mentioned above, Section 2.4.2 discusses systems for GWAS data analysis with a focus on unique genomic data management capabilities tailored for study constraints.

2.4 GWAS

In the middle of the 19^{th} century, Gregor Mendel made the first scientific discovery in heritability [84]. His observations of experimental work with pea plants summarised into three principles or laws, known as Medelian's laws:

The law of dominance

If a person received two different alleles from the parents and only one of the

alleles is contributing to the phenotype which is visible. There, contributing allele is called the dominant one for the selected phenotype.

The Law of Segregation

For any trait, each parent's pairing of genes (alleles) split and one gene passes from each parent to an offspring. Which particular gene in a pair gets passed on is completely up to chance.

The Law of Independent Assortment

Different pairs of alleles are passed onto the offspring independently of each other. Therefore, inheritance of genes at one location in a genome does not influence the inheritance of genes at another location.

These findings marked the beginning of genetic research, in which significant progress was made in the 20^{th} century [85].

Recent significant discoveries in genetic research include the following: (1) identifying the importance of DNA in determining genetic heritability [86], (2) identifying the structure of the DNA molecule [10], (3) understanding the genetic information encoded in the DNA molecule [87], (4) decoding the mechanisms for transforming information stored in DNA into protein structures [88], (5) introducing DNA recombinant technologies to generate synthetic DNA molecules in laboratories [89] and (6) automated DNA sequencing [90].

In the 1980s, attention turned to identifying genes in the human genome. The discovery of new genes was enabled by advanced technologies that allow gene mapping onto a chromosome and sequencing technologies for analysis of existing DNA structures. Based on new gene mapping and sequencing technologies, the term genomics has become prominent in modern genetic research [22]. These vast advances in genome sequencing and mapping have led researchers to develop new ways to treatment methods, drugs, or even prevent many diseases that afflict humankind [91].

Mapping the entire human genome is important because it helps us understand the instructions encoded in DNA and their effects on human health. Genomics projects from the 1980s have largely been successful and have led to following discoveries: (1) full sequencing of the bacterial genome [92], (2) the discovery of disease-causing genes by analysing inheritance patterns [93], (3) enhanced techniques (e.g. using the yeast genome [94] and the worm genome [95]) to identify the location of a gene on a chromosome and (4) the random shotgun sequence technique for DNA extraction [96]. These discoveries paved the way for the human genome project which began in 1990 and aimed to sequence the entire human genome. The sequencing of 99.99% of the human genome took nearly thirteen years [97] to complete.

In parallel with the progressive success of the human genome project, genomic medicine has gained significant momentum within the medical research community [98]. Studies in human genetics have long concerned in understanding, diagnosing, and treating diseases that show a clear and Mendelian (i.e., single-gene) inheritance [98]. However, many of the clinically diagnosed rare disorders and syndromes are good hypotheses of genetic inheritance, but not proven examples, of Mendelian inheritance [99]. The initial challenge in genomic medicine was to identify genes that cause or influence specific traits or diseases (that is, phenotypes) [100]. But until the 1980s, there wasn't a successful method to identify these relationships. In 1983, the first genetic marker for disease was discovered: the gene Huntington's disease [101].

Genome-wide linkage analysis was the first successful method for mapping disease markets to the human genome. Linkage analysis introduced a methodology to map gene loci to diseases in related individuals (e.g. families) [102]. Family-based Mendelian disease loci were identified using DNA polymorphism techniques [93]. The DNA polymorphism understands the different varieties of chromosomal locus situated in the same species population. This approach has successfully identified more than 1200 genes or loci on the human genome that are associated with dis-

eases and traits [100]. The first successful linkage analysis led to the discovery of diseases caused by the cystic fibrosis gene [103]. Linkage analysis was also used to identify the BRCA1 and BCRA2 genes for early-onset breast cancer [24].

Linkage analysis was used as the primary tool for mapping Mendelian and complex traits with familial aggregation for many years. However, this method has a number of limitations: (1) less helpful for complex traits, such as diabetes where multiple genes are important in disease causation [104], (2) genetic and phenotypic information is difficult to obtain for older generations [102], and (3) large family structures are required to understand complex diseases because the statistical significance is correlated with sample size [102].

GWAS are a modern alternative approach to linkage analysis. GWAS measure the genotypes of common genetic variants (markers) in people affected by the disease of interest (cases) and those unaffected by the disease (controls) [105]. These markers are referred to as SNPs. Typically, a GWAS measures hundreds of thousands of SNPs on few hundred individuals. GWAS compare the distributions of the marker in cases versus controls for to identify markers associated with the disease of interest. If any of the selected markers do not show significant association with the disease, they are removed from further assessments [106]. A GWAS may cost up to 10 million US dollars for its total expenditure due to the genotyping costs [46]. Therefore, researchers have to use the maximum outcome from the genomic datasets obtained for the studies.

The advanced technologies have contributed to developing affordable high-density SNP chips [107]. These SNP chips have allowed GWAS studies to be conducted. The foundation step of a GWAS study is selecting a SNP chip. The two major commercial SNP chip manufacturers are Affymetrix and Illumina. These chips developed by specialised techniques to extract the genomic information of biological specimens based on chemical analysis and electrophoresis. These commercial SNP chip manufacturers value their chips based on the genomic coverage for population sample and declare a price for the chip dataset [108]. As an example, a typical Illu-

mina chip consists of 500000 to 2500000 SNPs, but Affymetrix has limited the SNP count to around 1000000 by developing different chips for the different population groups, thereby reducing the cost of a chip. These strategies have contributed to defining the cost of a chip and researchers tend to select a chip that matches the original GWAS objectives.

2.4.1 GWAS analysis techniques

According to the findings of the HapMap project, approximately 500000 SNPs are sufficient to cover important genetic information in the non-African human genome [27]. Therefore, the selection of an appropriate population sample and extracting genotype information related to disease-causing genes is necessary to conduct an effective GWAS [109]. Less than 1% of the SNPs in the human genome cause phenotypic differences between the humans, and more than 1% of these SNPs cause genetically complex diseases [109]. Using large samples give more statistical power to identify rare alleles associated with disease [109].

When conducting GWAS analyses, strict quality control measures are required to eliminate false findings. This includes quality control of DNA samples and quality assurance for the common casual variants identified from the GWAS [110]. Common quality control steps include rejection of DNA samples that fail to meet standards (e.g. low quality DNA samples) [110] and rejection of SNPs with high missing call rates (e.g. SNPS with more than 5% missing calls will be rejected [110]). To ensure the quality assurance of a GWAS, researchers would consider the following: (1) check for gender misidentification in sex chromosomes [110], (2) check for relatedness in sample alleles [110], (3) test the Hardy-Weinburg equilibrium for allelic frequency [110] and (4) conduct principal component analysis to avoid population stratification by comparing the SNP allele frequencies similar to existing HapMap groups [111]. These common quality assurance methods are included in popular GWAS software tools such as PLINK and R [43].

Popular GWAS analysis techniques can be grouped into two major categories [112]. The first approach is to conduct a standard analysis to check the association of each SNP, one at a time, with the selected disease. The second one approach is to check the disease association for groups of SNPs (e.g. genes, pathways and intergenic regions). There are two popular statistical methodologies for testing single SNPs: contingency table and regression [111]. Using contingency tables, each SNP is tested for association with the disease; this yields a p-value for the strength of the evidence against the null hypothesis of no association with disease [111]. The chi-squared and Fisher's exact tests are largely used by GWAS researchers for this purpose [111]. The Fisher's exact test is recommended when the GWAS consist of a small number of samples.

Regression analysis can be used as an alternative to contingency tables to identify SNPs associated with the disease. Linear and logistic regression models are used to check each SNP's association with continuous and binary traits observed from the subjects selected for the study [111]. Regression models can also be used for testing groups of SNPS for association with the disease of interest [111].

Every GWAS must include a multiplicity correction regardless of whether it has been conducted to test associations for a single SNP or a group of SNPs. In GWAS, hundreds of thousands or millions of p-values are calculated to test the SNPs or SNP groups. Therefore, a false positive result is likely. If the false positive rate has been set to the 0.05, there is a 5% possibility of falsely rejecting the null hypothesis. One way of handling the multiplicity correction is the Bonferroni adjustment, in which the alpha value = 0.05 divided by the number of tests [111]. Another approach is to calculate the false discovery rate, which can be used to estimate the expected number of false discoveries [111].

SNPs that pass multiplicity correction and SNPs that have p-values less than the genome-wide significance of 10^{-7} are considered to be associated with the disease [113]. These results have been commonly presented by a Manhattan plot generated against the individual SNP positions in the chromosome and their p-values

(see Figure 2.3).

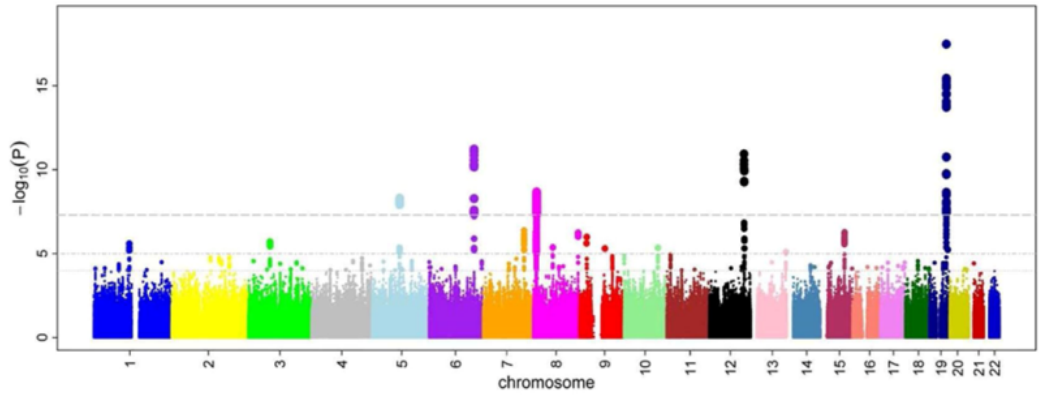


Figure 2.3: Sample Manhattan plot (reprinted from [1]).

2.4.2 Systems for GWAS data management and analysis

Section 2.3.2 discussed informatics approaches to biomedical data management and highlighted specialised systems developed for medical research and to manage existing phenotypic datasets. Genomic researchers have taken a similar approach and developed a specialised system to facilitate GWAS research.

PLINK

PLINK is a standalone open-source GWAS analysis package [43] that includes the majority of the analytical techniques discussed in Section 2.4.1. Specialised data management techniques developed for PLINK include storage file formats (PED/MAP and binary data format) and data management techniques (e.g. recoding existing physical data markers and updating SNP maps). PLINK file formats provide logical meaning to existing datasets and binary data formats save significant physical disk space. PLINK data management techniques have provided

the capability to dynamically update existing GWAS datasets without manually traversing the large GWAS datasets. PLINK analytical techniques cover a large spectrum of GWAS analysis including summary statistics calculations (e.g. missing genotype and phenotype rates and allele frequencies), association testing for the selected disease traits, comparing the allele frequencies between the cases and controls, and permutation procedures to calculate the significant empirical levels of GWAS results.

Although PLINK has implemented large proportion of analytical techniques for GWAS data, it has an execution environment running on a single computer. PLINK lacks parallel execution and cannot use HPC power to analyse large GWAS datasets. Therefore, researchers have had to implement parallel versions of PLINK using R, Stata, Matlab and MP, which are analytical software packages with programming interfaces supported by parallel analysis implementation. R is an open-source analytical package which consists of a server to deploy the individual R modules. Stata is a commercial statistical analysis package that has either a per user or group-based license. These licenses are quite expensive. Matlab is another commercial software package for statistical analysis and data modelling. Open MP is the parallel programming interface based on the C++ programming language. These packages provide unique programming guideline and operators to develop analysis packages. The parallel programming packages can be deployed to the expensive high performance hardware facilities uniquely operated based on separate distributed operating environments. One common aspect of these analytical tools and platforms is that they operate as standalone applications. Researchers have to develop separate data management mechanisms and connect to HPC sources by using command line tools. Researchers are also required to change and optimise the packages based on the HPC-supported programming framework and manually deploy whenever there is a change of commands.

Galaxy

Galaxy is a web-based biomedical data management and analysis platform for genomic analysis [45]. The Galaxy project operates under an open software license and is capable of functioning independently under different research administrations. Therefore, Galaxy users are able to customise the system interface for different HPC facilities and data storage infrastructure [114].

The Galaxy platform is capable of

- Managing data libraries, including navigation to a specific data source, viewing meta information related to a data source and importing a data source for the analysis.
- Running an analysis over one or more data sources or running multiple analyses on the same data set and filtering analysis results.
- Managing history and workflow imports. History management is saving the individual steps of analysis. A typical GWAS analysis consists of multiple steps that need to be run consecutively to complete a full analysis. Researchers can run these analysis steps according to a timeline and retrieve the results after completing the analysis. Galaxy can group individual analysis steps into a history and import total history as a summarized workflow at any given time. This helps to backup and import the existing histories to Galaxy and backup the workflows.
- Visualising results. Galaxy has an inbuilt visualisation engine to display the results. The visualisation techniques are shareable and can be applied to multiple result sets.

Galaxy provides a common platform to conduct genomic analysis in independent research infrastructures. Galaxy's workflow and visualisation import capabilities

help to share the analysis and results among the genomic research community. Results can be compared in a similar format and the same analysis techniques can be applied. Developing the analysis and visualisation techniques required a significant amount time cost. Galaxy saves researchers the development cost of specific techniques by enabling sharing of the existing techniques among collaborators. Galaxy can be integrated with HPC for analysis and compatible with the unique accessing technique facilitated by the HPC source.

GWAS Pipeline (GWASpi)

GWASpi is a multi-platform desktop computer GWAS analysis tool [115]. GWASPi caters for the data management aspects of the GWAS and analysis within the capabilities (processing and disk space) of the desktop computer. GWASpi is a Java-based application that operates inside the Java runtime environment and uses a Java in-memory database called Derby for application data management. GWASpi supports multiple GWAS sequence data formats including Affymetrix, Illumina and PLINK. GWASpi is supported by the high-dimensional data management library called NetCDF for real-time data management and processing.

GWASpi is capable of data cleaning and analysis. Data cleaning includes imputing missing genotypes per sample and per SNP, checking for allele mismatches and Hardy-Weinberg quality controls. Analysis includes calculating the allelic, genotypic and trend association tests. The results are output as reports and corresponding QQ and Manhattan plots. Given its open-source implementation, GWASpi enables researchers to customise its capabilities to meet individual research needs and to be suitable for the desktop computing environments. The capability of the system depends on the desktop computer memory, and it is volatile with the uptime of the desktop computer. Furthermore, it hasn't included a permanent data storage supported by a data management system and also lacked the HPC backend

which is quite necessary for the analysis of the large GWAS datasets.

2.4.3 Notable GWAS findings

According to the National Human Genome Institute web page[9], 35185 GWAS have been published to date. The number of published GWAS has increased rapidly in recent years, especially for diseases such as breast cancer and type II diabetes [116].

The first published GWAS examined age-related macular degeneration in the Chinese population [117, 46]. This study selected 96 cases who were previously diagnosed with age-related macular degeneration and 130 unaffected controls. DNA samples collected from the study participants underwent significant quality control checks and 97824 SNPs were selected for the analysis. The study found that a SNP in the HTRA1 gene on chromosome 10q26 has a significant genetic effect on age-related macular degeneration. A subsequent GWAS has identified two more genes (PLEKHA1 and ARMS2) that are associated with the risk of age-related macular degeneration [118].

The first large-scale GWAS was the Welcome Trust Case Control (WTCC) study conducted in 2007. The study considered seven common diseases: bipolar disorder, coronary artery disease, Crohn's disease, rheumatoid arthritis, and type I and type II diabetes [119]. This study had 2000 cases and 3000 controls. Cases were chosen such that each was affected by only one of the seven diseases. Subjects were selected from the British population and genotypes were measured using the Affymetrix 500K SNP chip. The study identified 24 SNPs associated with the risk of bipolar disorder, one SNP for coronary artery disease, nine SNPs for Crohn's disease, three SNPs for rheumatoid arthritis, seven SNPs for type I diabetes and three SNPs for type II diabetes.

GWAS have been conducted for many cancers including breast, prostate, colorectal, lung and melanoma [120]. Recent studies of genetic susceptibility to breast

[9] `http://www.ebi.ac.uk/gwas/`

Study	Cases	Controls	Genes
Genome-wide association study in East Asians identifies two novel breast cancer susceptibility loci [121].	7619	6286	*LMO4*, *LINC00160*
Genome-wide association study of susceptibility loci for breast cancer in Sardinian population [122].	1367	1658	*FGFR2*, *TOX3*
A pilot genome-wide association study of breast cancer susceptibility loci in Indonesia [123].	89	46	*SOGA2*, *CTNNA2*
Genome-wide association study of breast cancer in Latinas identifies novel protective variants on 6q25 [124]	1497	3213	*ESR1*, *TOX3*
Novel breast cancer susceptibility locus at 9q31.2: results of a genome-wide association study [125].	2839	3507	*SLC4A7*, *FGFR2*, *TOX3*, *KLF4*, *ACTL7A*, *RAD23B*, *ESR1*, *FGFR2*
Genome-wide association analysis in East Asians identifies breast cancer susceptibility loci at 1q32.1, 5q14.3 and 15q26.1 [126].	2867	2285	*ZC3H11A*, *ARRDC3*, *PRC1*
Genome-wide association study of breast cancer in the Japanese population [127]	2642	2099	*FGFR2*, *TOX3*, *LOC643714*, *ATF7IP*

Table 2.1: Recent breast cancer GWA Studies; sample sizes and identified genes associated with breast cancer

cancer have found significant associations with breast cancer and the LMO4 and LINC00160 genes [121]. According to the National Human Genome Research Institute web page[10], 35 GWAS have been conducted for breast cancer to date. The following table 2.1 summarises the most recent GWAS breast cancer and includes the sample sizes and significant genes identified in those studies.

[10] `http://www.ebi.ac.uk/gwas/search`

2.4.4 Beyond GWAS and towards personalised medicine

Two significant conclusions have been driven by the first wave of GWAS. First, highly accredited traits associated with certain genome positions have found to be related to many other positions in the genome [128]. Height is a typical example; a GWAS of 183,727 individuals concluded that is minimum of 180 genome positions contribute to height [129]. Second, even if all the locations in the genome that contribute to a specific trait are identified, an individual location's contribution to the overall trait variability is quite small [46]. Therefore, the missing heritability has not been discovered by the GWAS.

Scientists have debated the cause of the missing heritability and proposed rare variants, copy number variations, network effects, environmental exposure and epigenetic effects [128]. To try to understand the unexplained biological heritability, a Biology 2.0 approach was introduced to the scientific community [130]. It discusses the introduction of the HPC sources with an affordable cost. Furthermore, reducing the genome sequencing costs will bring much-advanced knowledge to the understanding of heritability. The Biology 2.0 era, aim to sequence large number of whole genomes and understand the existing patterns using the massive computational resources available for analysis.

The recent advancements in the informatics approach to managing big data, the availability of increased computational power for medical research and the decreases cost of genomic experiments allowed a new paradigm to emerge. This new paradigm has been called precision medicine or personalised medicine and involves treatment being tailored to suit an individual or group of individuals [131]. Personalised medicine was derived from population medicine where individuals are grouped into subpopulations based on their phenotypic and genotypic information.

Personalised medicine aims to equip physicians with prediction tools to support their preliminary judgments of a trait based on available patient genomic and phenotypic information. Precision medicine enables physicians to recommend the best possible treatment for the patient including much safer drugs and optimising a

drug's effect on the individual. Precision medicine helps to reduce medical costs by enhancing the chance of a drug helping patients, reducing the timeline for discovering disease and eliminating unnecessary treatments based on false assumptions.

The GWAS methodology is relevant to personalised medicine because it discovers similarities in the genomic code of individuals that are associated with phenotypic traits such as disease. The reducing cost of genomic enables more individuals to be tested. This helps researchers to understand a subpopulation's common ground in contributing to a specific trait. These subpopulations will share common treatment for the diseases which inheritor susceptible in their genetic code [132].

The increasing number of biobanks around the world has facilitated the much needed genetic information for precision medicine [133]. Biobanks that are well equipped and supported by software systems has increased capabilities for information retrieval on demand. The United States government has initiated consortiums to build practical guidelines for precision medicine (Collins2015). Supported by existing genetic knowledge, massive data warehouses store medical information data sets and through building new tools will be deliver an advanced medical platform for precision medicine.

The next section will discuss the HPC resources available to support GWAS research and ultimately, personalised medicine.

2.5 HPC

Advanced software systems are required to support GWAS research based on heterogeneous phenotypic and genotypic datasets of the big data scale. These systems enable data processing with complex analytical algorithms. HPC means aggregating the processing power of several computers and producing substantial processing capacity that is not available on a single computer [134].

2.5.1 Introduction to HPC

According to Moore's Law, the number of transistors able to be placed on a dense integrated microprocessor doubles every year [13]. That means the computational power of microprocessors is growing at an exponential rate. While computational power has dramatically increased, the emergence of complex applications and increasing dataset sizes often demands more processing power than is available from a single processor. To address the need for increased computational power, HPC has been introduced to aggregate the computational power of many processors and facilitate parallel processing. Today, HPC is available to a large user base in a variety of forms and computational mechanisms. In many cases this is motivated by the advancement of internet technologies such as e-commerce and social networking. Furthermore, advanced genomic sequencing technologies developed to extract whole genome sequences require massive computational power for analysis within a tractable amount of time.

The progression of technology has also seen numerous changes in approaches to HPC. In the 1950s, the development of the silicon transistor opened a new door to the computing industry by vastly reducing the size and cost of computers. The 1960s and 1970s saw little focus on desktop computers; instead, manufacturers focused on developing very large standalone computers. Vendors provided a propriety set of operating systems and programming languages (e.g. IBM's Fortran) and compilers to work on each machine. For specialised tasks, manufacturers built high-performance monolithic supercomputers to provide the necessary computational power. A supercomputer is an especially designed single computer that has significant processing power compared with a single desktop computer. These machines consist of an operating system and a programming interface that has been developed to support parallel processing.

Factors like affordability, size reductions and processing advances led to the introduction of desktop computers. Given factors has made a proportionate reduction in the supercomputer market and users seeking desktop based HPC solutions.

The shift towards desktop computers in the 1970s and 1980s, coupled with special application-level services, facilitated the emergence of clusters based on aggregating the power of independent commodity computers. These clusters, in turn, led to the development of grid computing, a method by for organisation-wide sharing of computing clusters [135]. Large-scale information technology companies such as Google and Amazon have met the challenges of the internet's rapid expansion by introducing the cloud computing concept. Cloud commodities have the ability to provide HPC as a dynamic service on a request-by-request basis. Today, cloud service providers are capable of providing on-demand, dynamically controllable, relocatable, fully managed, back-up and load balancing services to required parties.

Research in the field of HPC focuses on issues of performance, parallelism, control, latency management, namespace distribution and deadlocks [136].

Performance

The performance of large computational commodities is typically measured by floating point operations per second (FLOPS) conducted by a single computing commodity [137]. Starting with hundreds of kiloFLOPS [138], current supercomputers are capable of more than 93000 teraFLOPS of processing power according to the supercomputing Top 500 list [139].

Parallelism

Parallelism refers to the number of jobs that can be executed simultaneously by the HPC system; more tasks running in parallel will lead to a quicker execution time. Although massively parallel implementations can often be achieved on modern HPC infrastructure, programs are bound by Amdahl's law of parallel implementations [14], which states that parallelism might not be always applied for an entire problem and execution time is always bound to non-parallel computations.

Control

The architecture of HPC infrastructure may differ significantly, and systems that interface HPC platforms must have available a consistent method of control over these different implementations. This was a common situation when building the first supercomputer, Cray-1. Due to the scarcity of HPC sources, the system architecture, secure communication and operating system is always unique. Therefore, a single program has different versions for different HPC platforms.

Latency

Latency management includes the issues of communication bandwidth, multithreading, caching and message parsing with multiple processes [136]. There, consider causality for the communication delays in the program execution. HPC hardware architecture always consists of multiple processors, and their cores are executed simultaneously. Process execution in an HPC system consists of executing independent programs in parallel and later amending the results for a single decision. The final result depends on the latency management best practices followed during the program implementation. Also, depends on hardware support embedded in the execution environment to manage the latency.

Namespace distribution

Namespace distribution includes the individual label-based activities performed on memory management, I/O operations and process management [140]. Memory sharing policies are implemented in HPC systems to avoid deadlocks, special techniques are used to avoid the delays caused by I/O operations and parallel process management policies are adopted to optimise the application performance.

Deadlock

In a parallel programming environment, a deadlock is a scenario where the same resource is required by more than one processes at the same time. This

can be a problem in a parallel programming environment where multiple parallel processes run simultaneously to complete a task. If a deadlock occurs, threads can be hung in the wait state forever without knowing how to solve the deadlock. Most systems prevent deadlocks by following the ignorance, detection and the prevention procedures.

A key aspect to the success of HPC platforms is the availability of open languages (or language extensions) for parallel programming. The message parsing interface (MPI) was introduced in the mid-1990s to facilitate a common programming API for parallel programming problems [141]. The MPI specification includes multi-threaded operations, concurrent multiple process execution and fail-safe network operations. According to the MPI specification, the programmer decomposes a problem into parallel sub-problems, and then programs these as tasks that are distributed to the HPC nodes by the MPI. Then the results from those subprograms are concatenated by the MPI and output as an aggregate result set according to a predefined logic.

Open MPI [142] is the open-source implementation of the MPI specification, fostering a wide user community and providing a general implementation for handling distributed memory in parallel application architectures [142]. Open MP [143] is another open-source API developed to handle shared memory platforms. Open MP differs from MPI by avoiding explicitly partitioning the application data structures [143] and reducing the additional programming effort need to adopt for the parallel programming [143].

2.5.2 Types of HPC

In the early 1960s, Cray built the first supercomputer, the Cray-1, to satisfy the requirements of the time for high computational power [144]. Cray-1 marked the start of the supercomputing era by providing the computational power of ten individual computers of the time. The Cray-1 supercomputer was a custom hardware

platform comprising a single large unit or several large modules. Cray-1 had an operating system, multi-user synchronisation and a quota-based process allocation for an authorised set of users. Cray-1 proved to be a success for the company, selling hundreds of units, at a unit cost of USD 8 million [145].

Cray supercomputers dominated the 1970s and 1980s and Cray became the dominant supercomputing manufacturer worldwide. By the end of the 1980s, supercomputers could provide a processing capability of 333 mega-FLOPS per processor supported by the eight vector processors [146]. In the vector processors, instructions are held in a vector dataset, which adds significant processing capability to the processors. Backed by many governments and large organisations, supercomputing platforms advanced rapidly, and today there are supercomputers with many peta-FLOPS of processing power, including the IBM BlueGene series, Cray supercomputers and the Sunway Tianhe series [147]. The Sunway TaihuLight supercomputer achieves 93 peta-FLOPs of processing power [139]. Accommodating the supercomputer power for the various organisations including private and government sectors has significantly improved time-based performance for their lengthy tasks , however this process has introduced issues like high power consumption, vendor dependency, initial large capital investment and security issues.

As an alternative to an expensive single supercomputer, researchers proposed cluster computing. At present, the most common form of HPC is the commodity cluster, an architecture that can include hundreds or thousands of independent commodity computers working in parallel. The Beowolf cluster was an early example of cluster computing success; the machine comprised 16 motherboards with Intel x86 processors, each with 256 MB of main memory [148]. The cost effective nature and flexibility of HPC clusters have to led to these systems gaining popularity [136]. Large and ongoing decreases in the prices of commodity microprocessors, memory, disk storage and high-speed networks have prompted the choice towards cluster computing for many installations. The cluster concept has given a favourable choice of selection for the small parties who are looking for HPC solutions including less

initial capital investment, on-demand processing power and more flexible system architecture. There are known disadvantages, such as a requirement for dedicated maintenance staff, lack of confidence for continued processing power and limited network bandwidth.

General purpose computing on graphical processing units is operated as a special type of HPC environment. The existing parallel processing power of graphical processing units (GPUs) is used for parallel execution of computing processes and enables less overhead on the central processing unit [149]. The relatively small size of a GPU cluster brings a massive advantage over the cluster computing and supercomputing infrastructures. Compared with multi-threaded and cluster operating environments, GPU computing has shown a significant advantage in terms of FLOPS per watt performance. Considering the advantages of GPU computing in physical space management and the power efficiency of the operations, the major downfall is the limited programming capabilities [36]. Specialised programming experts are required to develop the parallel algorithms to execute in a GPU environment with the set of fixed operators.

Following cluster-based HPC, grid computing was the next HPC breakthrough. Inspired by the idea of power grids in the domain of electrical utilities, researchers developed new grid-based systems based upon on-demand supply of high computational power [150]. Essentially, grid HPC enables the collaboration of geographically separated HPC facilities to use their computational power for on-demand tasks, with collaboration between HPC facilities providing a common interface for access. A number of HPC facilities have been created by different research groups, and using the grid concept, it is viable to aggregate the idle computational power in these clusters and supercomputers [151]. The grid approach avoids the need for specialised computational support and maintenance when dealing with large computational power sources. Grid environments implicitly provide common technical tasks such as security, load balancing and application framework support [152].

The cloud has experienced an extremely fast rise in popularity and has quickly

become a 21st Century buzzword. Supported by large software companies such as Google and Amazon, the cloud concept is based on grid computing fundamentals but provides significantly more facilities than HPC applications [152]; cloud-based HPC can provide similar computational power to grid architectures, but in a more standardised way. Also, cloud architecture facilitates large set of hardware resources, a large number of virtual storage and processing power, and high network bandwidth. HPC Clusters can be created, grown and shrunk dynamically, and can be linked in a grid-like fashion across cloud facilities. Cloud-based HPC can be moved between data centres transparently and replicated and backed up easily. New hardware can be added to the cloud without downtime, meaning that its computational power can be ever-growing without interruption to its services.

A typical cloud environment will provide application developers with (i) infrastructure as a service, providing potentially massive computing and data storage facilities and (ii) platform as service, providing a deployment environment for applications. Essential software components for application development, such as security and authentication, is provided by the software as a service layer of cloud computing.

2.5.3 Strengths and weaknesses

State-of-the-art HPC systems are currently capable of providing peta-FLOPS of processing power. Computer experts predict that, given the current rate of hardware performance increases, HPC platforms will be capable of providing exa-FLOPS of processing power within the next ten years [135]. Today, HPC is commonly used in the defence, electronics, communication, medical and financial sectors [153]. The significant competitiveness in these sectors and the mission critical decisions made in these fields have required accurate and fast systems. Few examples for such scenarios include, the utilization of HPC systems to control the cooling system of a nuclear reactor or define the goods price in supermarket chain considering the

current demand and supply. Advances in scientific disciplines such as genomics, earth science and astronomy have generated massive datasets which mostly exceed the gigabyte scales. Analysing these datasets with complex techniques within a feasible time have required large HPC sources.

HPC platforms do face a number of challenges. Large-scale HPC infrastructure requires very large (to massive) amounts of electrical power for daily operation, and this, in turn, produces a significant amount of heat to be dissipated. Consequently, expensive cooling mechanisms are required to avoid hardware overheating, a problem that would lead to system instability. If an HPC system that requires one megawatt of electrical power for operations also requires 0.7 megawatts of electrical power for cooling [134]. For example, the Japanese Earth Simulator, considered the world's fastest supercomputer from 2002 to 2004, required 12 megawatts of power for its operations [134]. In many cases, the network bandwidth that joins HPC nodes is not in-line with the data processing speed of the processors, leading to a situation in which processors are waiting on network communications and are not fully used. Furthermore, a large capital investment is required to build and maintain an HPC facility, and issues of ethics and security arise when sensitive data (e.g. genomics data) is stored in a remote cloud environment.

Importantly, developing parallel applications for HPC platforms may incur significant costs [64]. In contrast with applications developed to run on a single machine, parallel programming is required to satisfy design and controlling constraints. Algorithms that are to be run in an HPC environment and gain the maximum processing power of the facility has to be implemented supporting the parallelisation. Optimum parallelisation will achieve the best use of the facility and speed up the processing time. As an alternative to parallel algorithms, developers can design different parts of the programs for independent datasets and execute these simultaneously. This also achieves the similar effect of parallel program implementation. Although techniques like MapReduce [34] have been introduced to parallel programming, developers have to deal with platform dependencies, licensing issues

and portability issues. In the application design, it is crucial to understand the parallel and non-parallel programming parts and program flow controls to avoid deadlocks and memory overheads. Programmers also need to update their code when the system architecture has changed, or code is ported to a new HPC environment. Code porting is a very common situation today because HPC is now enabled via various computational sources.

Although the availability of HPC has increased in recent years, interacting with the facilities has not changed much [154]. While programming languages have evolved and distributed computing frameworks have been introduced (see the previous section), they have to be compliant with the HPC architecture. Therefore, users must have multiple versions of the executing algorithms for the different computing facilities. Even if the executing algorithm has been parallel programmed, processing data has to be manually uploaded to the execution facility. Datasets used in HPC are typically categorised as big data and require substantial time and effort to upload the datasets. After that, a user has to build and deploy the parallel algorithm into the execution facility if it is not already configured. Executing a parallel algorithm in an HPC facility is the same as running an ordinary program. The user has to create an execution script specifying the runtime instructions (number of central processing units allocated for the execution, unique job id to monitor the program, etc). After the execution script has executed, the user can monitor the runtime progress by logging into the system and extracting results when the job has completed. For further analysis, users need to examine the results and make a decision on future steps. These steps are quite feasible for computer scientists with their theory and practice in similar environments. For other researchers, this has created considerable overhead in their day-to-day activities and they often need specialised support from outside parties.

2.6 Conclusion

This chapter has reviewed the application of data management technologies, first in the generic sense, and then as applied in the field of biomedical research. The review has described the evolution of data management approaches and the technological innovations and semantics of the datasets. This was followed by a discussion of the introduction of modern database systems based on the relational data model and its sustainability for normalised datasets. The introduction of the internet has created extremely large unstructured datasets required development of NoSQL databases specialised to handle big data.

The internet era has made a significant impact on human life and existing research best practices. The introduction of the web-based software systems had a huge impact on data storage in research studies. For medical research, biomedical informatics has introduced a new data management paradigm for existing medical research datasets and sophisticated interactive web-based tools to manage these datasets. Importantly, while advanced web-based systems such as The Ark and REDCap have been developed, there is no system that combines conventional epidemiological research study data (e.g. consent, contact and questionnaire data) with intelligent pedigree modelling, or GWAS data management. The review further highlights that there is no system that integrates comprehensive data management of this type with suitable analysis platform in a user-friendly manner.

Next, the chapter reviewed the GWAS methodology as a popular tool for genomic research. Section 2.4.3 showed that GWAS has led to a number of notable medical discoveries.(e.g. the Welcome Trust Case Control study conducted for the seven common diseases). While tools such as PLINK and R software packages help researchers undertake a GWAS, there is no solution that seamlessly integrates the functionality of these tools with HPC platforms and providing semantic data management approach to GWAS.

Finally, a discussion of HPC systems was presented in Section 2.5. HPC plat-

forms fall into five broad categories: supercomputers cluster computing, GPU computing, grid computing, and cloud computing. At the time of writing, leading HPC platforms can achieve 93 PetaFLOPS performance, and are therefore well suited to support GWAS analysis. There are, however, several drawbacks, notably the lack of integration of these facilities with popular biomedical data management systems, in addition to the specialised computer skills required to use an HPC system to its potential. This poses a substantial challenge to epidemiologists who do not typically have the advanced computing expertise required to use these platforms.

The Ark: a customizable web-based data management tool for health and medical research [30].

3 The Ark

3.1 Introduction

This chapter discusses the nature of biomedical research datasets and the functionality of the information systems required to manage them. The introduction of computer-based software systems to manage biomedical research datasets has enabled a new research field called biomedical data management.

The Ark is an open-source web-based system designed to manage biomedical research datasets. Development of the Ark was initiated by researchers at The University of Western Australia who have considerable experience in managing biomedical research datasets. Staff at the Centre for Epidemiology and Biostatistics at the University of Melbourne began collaborating with the Ark developers in

2012, and today act as the Ark's primary development stakeholder. This chapter examines the Ark's architecture, supported technologies, and system build and deployment process. Project management is crucial to a successful software project. Therefore, this chapter includes an overview of agile project management practices for the Ark and source code management with collaborators of the project.

This chapter discusses the Ark modules by describing the objectives, functionality and expected outcomes. This is followed by a description of the current developments of the Ark and module enhancements including a critical evaluation of the modular architecture chosen for the Ark. This chapter then provides a set of use cases for the system by introducing the projects for which the Ark has been chosen as the data management solution and providing a description of the characteristics of the data collected by those studies.

Security is a crucial aspect of biomedical data management. The Ark has implemented user role-based security architecture to protect datasets. The chapter has included security discussions on authentication of the Ark users and how the Ark authorises the user activities. Furthermore, provides a detailed description of the Ark's security architecture, security framework and the secure deployment environment.

This chapter then compares the features of the Ark with other biomedical data management systems. This includes an introduction to existing biomedical data management systems and a discussion of how they manage an existing biomedical dataset. This chapter concludes with suggestions for enhancements to the Ark in its progress towards family and genomic data management.

3.2 History of biomedical data management and implications for the progress of biomedical research

Data-driven research has provided a pathway to many ground breaking research findings in medical science [21]. One of the earliest dates back to the 17^{th} century when sailors suffering from scurvy were treated with fresh citrus juice [155]. At the height of the London cholera epidemic, Dr John Snow observed the location of patients' residences and their water consumption sources [156]. Analysing the data using a set of dot maps, Dr Snow discovered a direct relationship between the contaminated water source and cholera patients who have consumed them. Identification of the contaminated water source helped the authorities to contain the cholera epidemic.

In the 20^{th} century, data recording standards evolved and technological innovations enabled advances. Observable changes (height, weight, colour, etc.) in research subjects (humans, animals and plants) and phenotypic information has been recorded and preserved in data management systems. Previously stored research datasets have been used across generations by medical researchers for various experiments [47]. Biomedical dataset management and its implications has been discussed in the Literature review section (See Section 2.3). Datasets include paper-based documents, databases and data management systems. Large amounts of data are generated by the medical research community at universities, hospitals and clinics. At the end of the 20^{th} century, complete sequencing of the human genome introduced genomic data to the medical domain. Closely observing the use of data in biomedical research, and understanding the capabilities of information management systems, the 2004 United States Food and Drug Administration annual report highlighted deficiencies in biomedical data management as a factor that may negatively affect medical discoveries [29].

The Food and Drug Administration report discusses a comparative decline

in significant medical research findings after the 1950s when computers and related technologies were introduced. As mentioned by the report, even though multiple computer-based data management approaches were available, most medical researchers preferred old-fashioned paper-based data management approaches. Around that period, medical practitioners preferred to write prescription by hand and research was conducted based on paper based material. The Food and Drug Administration report suggests maximising the use of computational power in biomedical research. The introduction of electronic medical records and the development of biomedical data management systems were key proposals included in the report. Driven by the report's findings, various medical research groups started working on developing data management systems. The increase in intention to manage existing biomedical datasets have further increased demand for the data management system developments within the biomedical research community.

3.3 Western Australian Genetic Epidemiology Resource (WAGER)

The Centre for Genetic Origins of Health and Disease (GoHaD[1]) at the University of Western Australia has collected datasets from many biomedical studies. Currently, the Western Australian DNA Bank and the Western Australian Research Tissue Network are the primary research assets in the Centre for Genetic Origins of Health and Disease's data vault. Driven by the obligation to efficiently manage datasets and the need to build biomedical data management systems, the Centre for Genetic Origins of Health and Disease developed its data management solution. The design objectives of this project included a centralised system update mechanism, no download requirements for new users, platform independence for users and browser-based access to the system. The system's functional specification consists of remote study data management (locally and internationally), work in different

[1] `http://www.gohad.uwa.edu.au`

time zones without limiting to the local area and enabling a set of user-friendly web interfaces to work with the system interactively.

The Centre for Genetic Origins of Health and Disease chose the proprietary application development platform Oracle Application Express [157] to build WAGER. WAGER was web-based and enabled remote researchers to log in via the world wide web. The development platform was selected due to its rapid application development capabilities and direct support of the Oracle database. WAGER was built based on the architecture provided by Oracle Application Express and following the best practices for platform development. WAGER was designed to hold data from multiple studies with subjects attached to each study. WAGER enabled researchers to navigate by selecting a study and a subject associated with the study. Selection of a study restricted the data view to a single study and operations were enabled only for selected study elements. Making the selected study into context increased the system security by restricting users to specific studies and organising the study data more reliably.

Successful completion of WAGER enabled researchers to migrate their datasets to WAGER to take advantage of its functionality. WAGER included management of the study, subject, phenotype and biospecimen data. WAGER also provided appropriate security control over data and enabled researchers to generate reports based on requirements driven by individual studies. WAGER remained with the Centre for Genetic Origins of Health and Disease around 10 years and held more than 25 study datasets belonging to different research groups (See Section 3.11).

As time passed, many studies were migrated to WAGER, and each brought a unique set of changes to the system. By 2010, most of the original WAGER developers had left the Centre for Genetic Origins of Health and Disease and there were significant costs related to recruiting specialists of Oracle Application Express development. Furthermore, the Centre for Genetic Origins of Health and Disease was faced with annual renewal of the platform license which required a significant financial outlay. Increases in system maintenance costs encouraged the decision-

makers to disband WAGER and find an alternative. Given situation has marked the beginning of the Ark project and developers started working on a new system based on open-source technologies.

3.4 The Ark

In 2009, the Centre for Genetic Origins of Health and Disease decided to implement a new system based on popular open-source technologies developed under the Apache license[2]. They named this system the Ark and used the existing WAGER functionality to plan the development and guide expectations. The Ark[3] currently operates as a leading open-source biomedical data management solution for the medical research community. The Ark supports more than 30 medical research studies operating locally and internationally [30]. The system is capable of handling multiple aspects of medical research data management (e.g., study management, study participant management, questionnaire data and laboratory sample management) with its compound web interfaces. A scientifically organised set of web user interfaces provides generic functionality to manage medical research data. At this level, the Ark was mainly focused on managing phenotypic datasets (e.g., contact, consent and biospecimen).

3.4.1 Architecture and implementation

Similar to WAGER, the Ark was designed as a web-based system. An N-tiered application architecture was selected and multiple (1 ... N) application layers have been declared to group similar application processes. Each application tier specialises in one or more application processes: (i) the web tier supports the web user interfaces of the application, (ii) the service tier supports the business processes and

[2] `https://www.apache.org/licenses/LICENSE-2.0`
[3] `http://sphinx.org.au/the-ark`

(iii) the data tier enables data-related operations such as querying and transaction handling.

3.4.2 Modular architecture

Due to its modular architecture [158], the Ark it is inherently capable of maintaining distinct independence between the application modules. The application start-up, security and system navigation modules are all managed by the top-level Ark-Container module. The Ark-Common module provides shared functions to all other Ark modules. Other modules such as Ark-Study and Ark-LIMS have individual functionalities but are implemented using the abstract features defined at a higher level in the Ark-Common module.

This type of modular architecture has helped to maintain a consistent codebase and coding style across the Ark's modules and re-uses features implemented in the Ark-Common module. Maintaining a set of well-tested components has the added advantage of increasing the system's stability and saves substantial development time from not having to duplicate features across modules. The modular approach helps separate research groups to continue their development activities independently. The codebase draws upon a single common security scheme that is implemented in all of the Ark's modules and maintains a common user interface design.

When the Ark project was initiated in 2009, the Java[4] programming language was the language of choice for web application development. Therefore, Java was chosen as the primary development language for the Ark. In addition to its popularity, large numbers (more than 100) of open-source application development frameworks were implemented based on Java by different developer groups. The Ark's founding developers chose to implement open-source frameworks for the initial application architecture.

[4] `www.java.com`

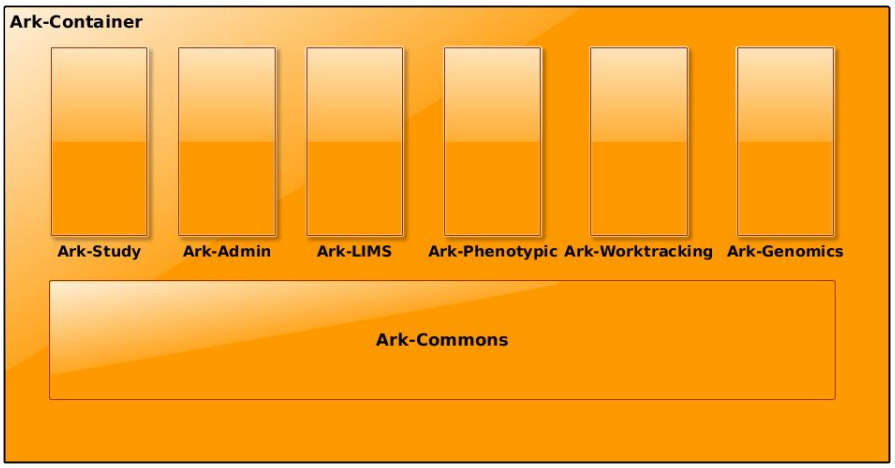


Figure 3.1: High-level view of the Ark's modular architecture.

3.4.3 The Spring framework

The Spring [5] is a Java framework for enterprise-level application development. The Spring framework has eliminated deployment dependencies and has enabled infrastructure support at the application level. Deployment dependencies include the application's runtime environment and third party libraries. The Spring framework hides wiring problems in communication among various tiers of the application, leaving developers free to focus on application logic. Therefore, developers can leave the burden of application memory handling, threading and architectural design patterns to their base framework.

Architecture best practices are inherent in the Spring framework including dependency injection, aspect-oriented programming, restful web services and support for various data integration frameworks (e.g., Hibernate and Enterprise Java Beans). Dependency injection creates reference objects when needed by the application framework. Aspect-oriented programming is used to annotate predefined operations available in the application framework (e.g., database concurrency handling and session timeouts), while restful web services operate as a communication mechanism between independent applications, and data integration is the mapping

[5] `www.spring.io`

between application and database tables.

The Spring framework has enabled an enterprise-level foundation for the Ark. The Ark project's use of the Spring framework's modular architecture with dependency injection allows independent application modules to be built from functional specifications and later integrated. Minimising dependency between the Ark modules has given much freedom to developers. The Spring framework enables a deployment-neutral development environment for application developers and lets them choose their deployment environment. Spring is a dynamically growing application framework and it contains a set of plug-ins to other frameworks.

3.4.4 The Hibernate framework

Software systems use relational databases to store datasets. The Ark has preconfigured application frameworks to match application object properties with existing tables and attributes. For object–relation mapping, the Hibernate framework[6] is considered the best data integration framework for Java applications. Hibernate eliminates data type mismatches between the Java model object properties and database column types. The Hibernate framework matches Java objects to database tables and acts as an intermediate to match Java data types with Structured Query Language data types. Hibernate has a query-building mechanism for information retrieval from database tables and maps the results to Java objects. Hibernate supports a wide range of databases via the native Java DataBase Connectivity driver that decouples the application data layer from the backend database.

The Ark has benefited from decoupling the application data layer from the supporting databases. Managing a database is not a simple task and requires a specialised set of skills. This decoupling has enabled researchers to choose their preferred database engine (open-source or commercial) to work with the Ark. The Hibernate query building mechanism has helped to transform complex queries to a

[6] `www.hibernate.org`

simple set of Java methods and has helped build the Ark's search criteria.

3.4.5 The Apache Wicket framework

Apache Wicket[7] is a component-based Java server-side framework to support web application development. Features of the open-source Apache Wicket framework include (i) simplicity, (ii) secure application development and (iii) support for asynchronous JavaScript and Extensible Markup Language technologies. The most important attribute of the Apache Wicket framework is its component-based design in which HTML elements are mapped to Java objects using namespace matching. This matching helps web application developers to implement presentation logic as a high-level Java implementation. Markup changes or web interface styles can be separately managed without conflict with presentation logic. The Apache Wicket application contains one-to-one matching of Java classes and HTML components.

The Apache Wicket framework has helped create a common web template for the Ark. This has enabled the Ark's developers to re-use web templates to build their functional user interfaces. The Apache Wicket framework gives a dynamic look and feel to the system. Its inbuilt internationalisation facility enables multi-language support for the Ark. As an evolving web framework, the most recent release of Apache Wicket (Version 7) provides an up-to-date set of web components for application development. The dynamic evolution of the Apache Wicket framework helps to match modest user interface requirements with the Ark (e.g., sortable data grids and dynamic calendar bookings).

3.4.6 MySQL database

MySQL[8] is a free open-source relational database management system that has been adopted by many open-source projects due to its feature rich implementation.

[7] `www.wicket.apache.org`
[8] `www.mysql.com`

MySQL has capabilities that are equal to a commercial database management system. MySQL supports many commercial database management system features including American National Standards Institute SQL-99 standards and extensions, cross-platform support, stored procedures, triggers and cursors. The American National Standards Institute SQL standards have been introduced to ensure that MySQL database operations are compliant with the same set of rules as those followed by commercial database vendors. MySQL supports a cross-functional facility by operating in different operating systems (e.g., Windows, Linux and OsX). MySQL supports database programming techniques such as stored procedures and cursors to define the amalgamated table functionality. Also, execute triggers on predefined conditions during the table data updates.

The Ark project has chosen MySQL as its data storage engine due to its popularity, rich features and its open-source GNU General Public License. MySQL supports the Ark's secure data warehousing requirements, simultaneous job execution, complex queries and database scalability. Importantly, MySQL enables the research community to use the Ark without requiring commercial licenses. MySQL's cross-platform support has allowed data managers to host the Ark on their preferred platform. In theory, any relational database management system can be used as the database for the Ark with a minimal workaround to the application source. However, the track record of the MySQL database with the Ark operations has proved its consistency and reliability.

The Ark is based on the Spring, Hibernate and Apache Wicket frameworks. The Ark developers have followed the best practices specified for those frameworks and carefully designed a modifiable application architecture. The security mechanism for the Ark is based on the Lightweight Directory Access Protocol (LDAP). Further details on the Ark's security will be discussed in Section 3.12.

3.5 The Ark's build and deployment

Build automation is considered best practice for software projects. Rather than manually compiling and building programs, pre-configured build tools allow developers to automate these procedures. Specialised build tools for software development help programmers to implement code and also help system administrators to install the software on the host. Once the builds tools have been configured, they can be run consecutively after making a change to the system. If it is a successful change, the build tools will re-build the system and the changes should be visible in next run. Any issues are immediately reported in the build logs and will require developers to re-run the build process after fixing the problem. Modern build tools are capable of compiling, executing test cases, packaging and even deploying to the runtime environment. For Java-based software developments, Ant, Maven and Gradle are popular build tools.

3.5.1 Maven

Maven[9] is a popular build tool for Java projects. To declare a build process and project dependencies, Maven uses an Extensible Markup Language build file. This build file describes the compilation, packaging and specific build tasks. Maven has a set of plug-ins to support the different build tasks, and users need to declare these so that they are understood by the tool. Dependencies are automatically checked against the Maven local cache and if not found, are automatically downloaded via the internet from a publicly accessible repository.

Maven provides a common build platform for the Ark and developers do not need to download and configure dependencies manually. The Ark was developed as a set of independent modules with individual dependencies, and each module contains its own build file. The Ark's main build file builds all modules and packages the system into single web archive. Its cross-platform support enables a neutral development

[9] `http://maven.apache.org`

environment for developers.

3.5.2 Apache Tomcat web server

Apache Tomcat[10] is an open-source Java web application server developed by the Apache Software Foundation. Tomcat is popular Servlet engine among Java web application development community because it is easy to install and manages server instances. The Tomcat web server was implemented to support the Java Enterprise Application specifications including Java Servlet, Java Server Pages, Java Expression Language and WebSockets. Therefore, Apache Tomcat is a servlet engine that is capable of hosting Java Servlet web applications. Apache Tomcat is quite popular in the Java web application community due to its open-source community support, load balancing capabilities, high availability, trouble-free server administration and multi-platform support; Apache Tomcat currently supports popular operating systems including Windows, Ubuntu and CentOS.

The Ark project is currently deployed in the Apache Tomcat web server. Even though the Apache Tomcat was selected as the preferred web server, the Ark can deploy in any Java application server that supports the Java Servlet and Java Server Pages specifications.

3.6 The Ark web based controls

The Ark is highly configurable and customisable via its web-based user interfaces. Based on the security principal of accounts and user groups [159], the Ark is presented as a set of functional modules to researchers. The study administrator or the system administrator can select which modules will be used for a study's data management and can configure how access privileges are granted to individual study users. The Ark's secure web controls and the access roles and functions assigned to

[10] `http://tomcat.apache.org`

them will be discussed in Section 3.12. The Ark also enables researchers to declare custom data fields and group them into categories based on semantic similarity

3.7 Project management

Being a collaborative project with developers at multiple sites, the Ark requires coordination and project management. Good project management methodology reduces the risk of failure in the software development process [160]. In light of this, the Ark employs agile software project management best practices [161]. The agile project management philosophy was introduced in response to failures that occurred using traditional waterfall methodology [162]. Traditionally, software development is a set of independent processes that operate sequentially. The typical waterfall model (see Figure 3.2) consists of the following phases:

Requirement analysis

This is an analysis of the software requirements and documentation of the requirements according to user perspectives.

Software design

This includes developer discussions on the technical representation of the requirement analysis and development of an architectural design for the software implementation.

Coding

This is the implementation of a software system based on the previously prepared requirement specifications and design guidelines.

Testing

The implementation of the coding phase is tested and the system functionality is verified to match requirement specifications.

Operation

The final stage of the software development project includes deployment, usability and system support.

The waterfall model would work fine in a textbook case of software development, yet that is not the reality.

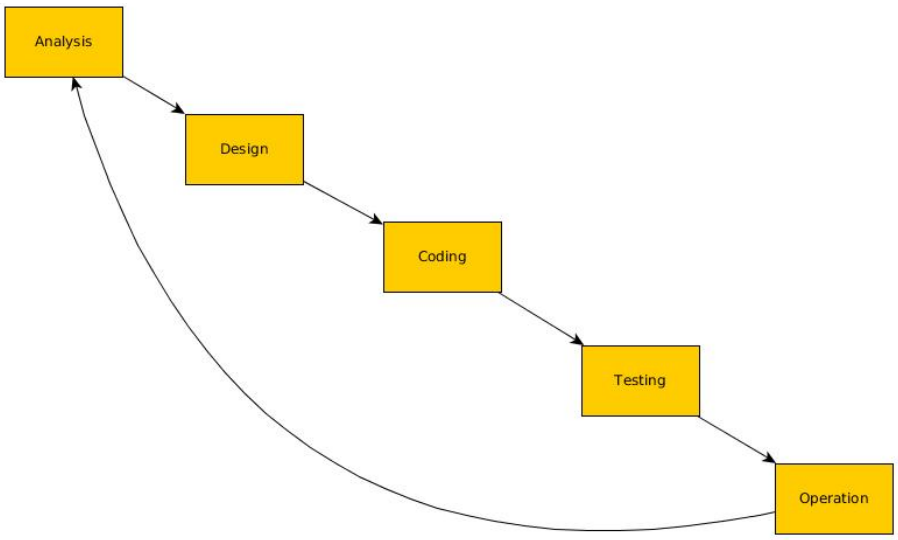


Figure 3.2: Traditional waterfall method based software development lifecycle.

Many software project failures are due to changes to requirements during the development phases and a lack of communication between the parties involved in different phases of the project cycle. In 2002, a team of software experts defined 12 principals in agile[11] software development. Customer satisfaction is the primary objective of an agile software development project. In addition to customer satisfaction, an agile project consists of: (i) frequently accepting changes to requirements, (ii) continuously delivering working software, (iii) maintaining close collaboration with developers and end-users, (iv) trust in the developer's capabilities, (v) encouraging face-to-face discussions, (vi) software workability as a measure of project success, (vii) parties working on the same phase throughout the project, (viii) being available to improve the technical quality of the product, (ix) simple design,

[11] `www.agilemanifesto.org`

(x) using team-oriented architectural best practices, (xi) regular reflections on the quality of work and things to be improved. Based on these principals, the Ark development team is keen to follow the agile principals for software development.

3.7.1 Agile Kanban project management

The Ark's development team follows agile Kanban methodology [163]. Based on the principles for an agile process, Kanban focuses on just-in-time software delivery without overloading developers. Kanban's team focus is on current work-in-progress items that are known as user stories. A user story is a work item in the software development process, such as enhancement, bug fixing or a deployment task. User stories are prioritised according to a common agreement between end users and developers. The entire collection of unfinished user stories is called the backlog of the project. Developers choose the user story with the highest priority. When a user story is completed, it is moved to the completion log and the developer picks the highest priority user story in the backlog. Completing a set of user stories is referred to as a single cycle, at the end of which a new software version is released. Within each cycle, developers decide the user stories, technical improvements and drawbacks. The workflow of a Kanban project is summarised in a column-oriented workflow call the Kanban dashboard (3.3).

To manage the agile process, the Ark uses the Atlassian JIRA system[12] to document system enhancements, bugs and improvements. Each JIRA story (A short description of a task) includes the description, reporter, assignee, current status, progress and metadata. After completing a development cycle, the Ark management team prepares a new system release. The following is an example of the development workflow. First, a user story with an implementation description is created in the JIRA system and assigned to a developer. The initial story status is Open and when the assignee has accepted and started the user story, its status is

[12] `https://the-ark.atlassian.net/secure/Dashboard.jspa`

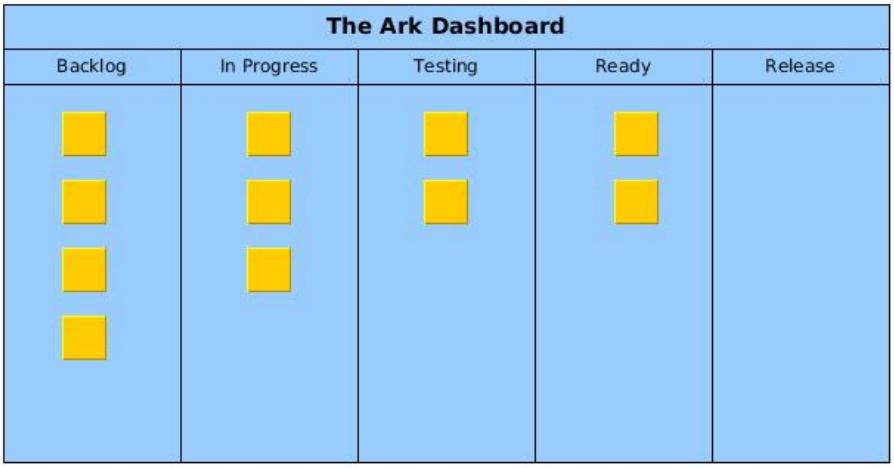


Figure 3.3: Sample Kanban board use to manage the user stories of software system.

changed to In progress. When the assignee has completed the user story, the status is changed to Test ready and assign to a tester. When the tester has completed testing and is satisfied with the implementation, the user story status is changed to Completed. When there are enough completed user stories in the JIRA dashboard, system administrators can then decide on a new release date.

3.8 The Ark source code repository

Large software systems are developed collaboratively. In some cases, developers may be geographically distant and need to simultaneously edit the same source file. To avoid conflicts, revision control systems are introduced to the development environment. Revision control systems are capable of keeping source files in secure cloud storage without losing the data in the case of a local developer's machine failure.

Large software systems are developed collaboratively. In some cases, developers may be geographically distant and need to simultaneously edit the same source file. To avoid conflicts, revision control systems are introduced to the development

environment. Revision control systems are capable of keeping source files in secure cloud storage without losing the data in the case of a local developer's machine failure.

Recently, the Ark's project source code has been moved to the GitHub repository[13]. There, the Ark uses the GIT source code management system to manage the source codebase. The GIT source code management system is popular in the open-source software development community due to its speed, integrity and inbuilt capability to handle distributed non-linear source repositories.

Anyone who has an interest in the Ark's developments can clone a copy from the main source code. There, the Ark committer can push back their changes, and others can maintain their repositories locally. The Ark committer's status is granted based on coding eligibility. The GitHub repository consists of up-to-date source code, and tag releases made for production and test purposes.

3.9 Collaborators

In 2012, the Centre for Epidemiology and Biostatistics[14] at the University of Melbourne joined the Ark project as a collaborator and started contributing to the Ark development process. Both centres (Centre for Genetic Origins of Health and Disease and Centre for Epidemiology and Biostatistics) have a reputation for maintaining large biomedical datasets in their data repositories and were eager to take advantages of the Ark to manage their datasets. The Centre for Epidemiology and Biostatistics development team introduced the Ark-Work Tracking and Ark-Genomics modules to the Ark. In 2014, the Lions Eye Institute[15], affiliated with the University of Western Australia) started using the system and also joined as a collaborator of the Ark. Initially, the Lions Eye Institute helped to resolve the existing user stories recorded in the JIRA system and started migrating their datasets

[13] `https://github.com/The-Ark-Informatics/ark`
[14] `http://epi.unimelb.edu.au`
[15] `https://www.lei.org.au`

to the Ark. Initial success in data porting encouraged the Lions Eye Institute to start developing the Ark-Disease and Global Search modules.

In addition to the Australian collaborators, international organisations are now using the Ark to support their biomedical data management activities. The Human Heredity and Health[16] in Africa Initiative has chosen the Ark as a cost-effective data management solution and has configured the Ark independently using the documentation included in the codebase. The Human Heredity and Health in Africa Initiative is a works on common diseases in African populations. Seoul National University's Graduate School of Public Health[17] has a large set of biobanking samples and subject demographic datasets. They have chosen the Ark to manage their biomedical datasets that include studies of breast cancer and colorectal cancer.

3.10 The Ark functionality

A study dataset typically consists of subject-related sub-datasets collected during the study period. Three tiers to a subject-based study dataset can be identified: study, sub-study and subject. Sub-studies are represented as a subset of subjects from the main study. Sub-studies generally have a more specialised research focus than the parent study. When designing a biomedical data management system, the range of sub-study management operations that required for the overall study need to be accommodated. Therefore, biomedical data management systems need to be as flexible and configurable as possible.

The Ark was designed to manage multiple study datasets in a single installation with different levels of access granted to researchers depending on their research programs. The need to accommodate multiple study datasets has driven the Ark development team to introduce the concept of study and subject context. To access data, the Ark user first needs to select a study in context. The user can then

[16] `www.h3africa.org`
[17] `http://health.snu.ac.kr/en`

access the study demographics, study components, custom fields, study users, bulk uploaders and reporting. Selecting a subject is called bringing the subject in context. The subject in context enables the researchers to work with subject-related datasets.

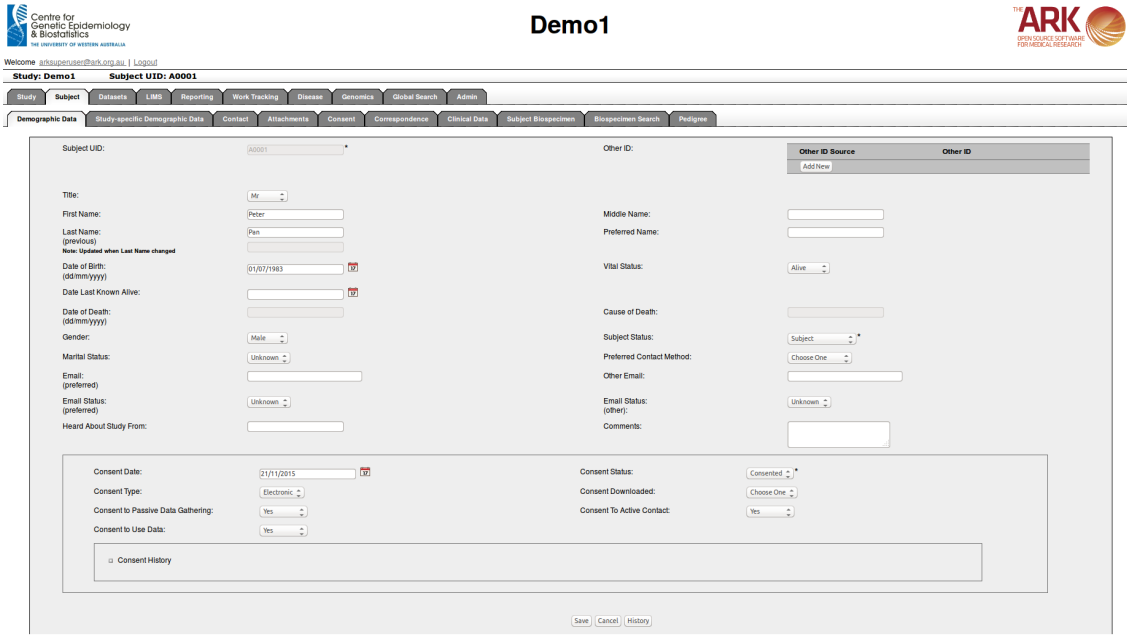


Figure 3.4: A subject demographic screen of the Ark, on top left side of the figure shows the study in context (Demo1) and subject in context (A0001).

The Ark-Study and Ark-Subject modules are the primary container for a study dataset. In addition to these modules, the system provides The Ark Dataset module for storing questionnaire datasets and phenotypic information, The Ark LIMS for storing laboratory data, The Ark Work tracking for tracking cost-based activities, and The Ark Reporting and Data Extraction modules to produce data summaries. The main contribution of this thesis is a genomic module that has been designed to support genomic data management and analysis.

3.10.1 Ark-Study module

The Ark-Study module was designed to initiate and maintain information about a study dataset in the system. The Ark-Study module allows configuration to meet a study's data management needs. The Ark-Study module includes following sub-components: (i) study details, (ii) study components, (iii) configuring custom fields and custom field categories, (iv) creating system users and assigning user roles to them, (v) bulk uploading subject data to populate a study, and (vi) global search. Custom field categories are used to group the custom fields and enable to create n-level subcategories. These help to organise custom fields into logical groups and optimise the visibility of custom fields.

Study details

Study details describe the properties of a study. Study identifiers include the name and description of a study, names of investigators, start date and other information. Defining the study status (as active, discussion, expression of interest, etc.) reflects the current status of a study dataset. Study administrators can decide which of the Ark's modules are to be enabled for the study users by selecting these in the study detail section. Tabs for modules that are not selected are hidden from the Ark user view. When creating a study, an administrator can decide whether this study is a primary study or a sub-study. A sub-study inherits the characteristics of the primary study including the subjects and related properties. A subject's unique identifier (UID) is the visual identity assigned to a subject. Study administrators can enable auto Subject UID generation for the study participants using a predefined text pattern.

Study components

Study components are the recurring events in a study's lifetime. For study events (e.g., blood collection or interviews.), study components records the properties

(name, description, etc.) of the event. When a study component has been executed for a subject will record against the subject consents (see Section 3.10.2).

Configuring custom fields and custom field categories

Custom fields and categories have added much more flexibility to the Ark compared with other biomedical data management systems. The Ark consists of a fixed set of demographic fields to record the subject information and it is capable of assigning custom fields to record additional information belonging to a subject and family. Custom fields can be used to group fields to semantically similar custom field categories and allowing researchers to manage character, number and date data types. Custom fields can be given properties such as required, multiple selection options for character types and default values.

Creating system users and assigning user roles

The study user section facilitates creating the study users an assigning them with specific user roles. These study users can access the Ark using their login credentials and are able to navigate to assigned study datasets. The processes for creating users and assigning specific roles to their accounts are discussed in Section: 3.12.

Bulk uploading the subject data to populate a study

The Ark has a set of bulk data uploaders to populate data records belonging to the study and its subjects. At the study, there are custom field, category uploaders and subject uploaders. At the subject level, there are custom field data, consent data, pedigree data and attachment data uploaders. The Ark's users download the data templates provided by the system and populate the templates with their datasets. An inbuilt set of custom rules validates each uploader, and when dataset passes validation, bulk upload can proceed. When the validated dataset has been

submitted for uploading, it will execute as an internal batch process and will not affect the functionality of the system. When the dataset upload has completed, this will be indicated as a dataset record in the bulk uploader.

Global search

The latest addition to the Ark-Study module is global search. The global search function is able to navigate through the Ark's study list and search for subjects and biospecimens without limiting to a single study. Global search helps system administrators to search across the system and select a specific item (subject or biospecimen).

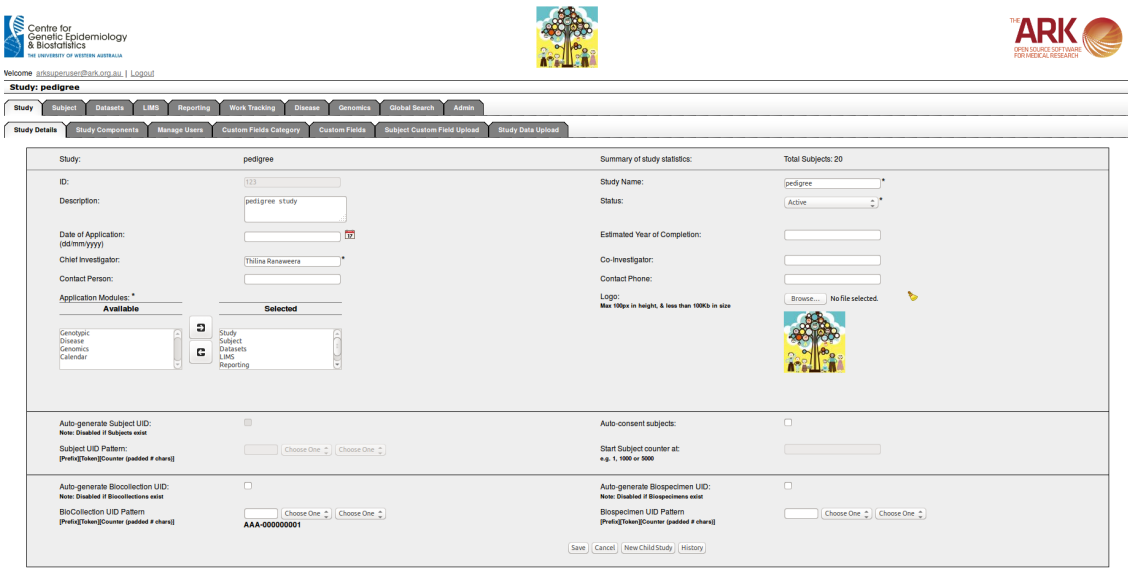


Figure 3.5: The Ark study detail screen with study Meta information and selected modules for the study.

3.10.2 Ark-Subject module

A study dataset typically comprises many subjects with data fields for each subject. The Ark-Subject module is capable of storing (i) demographic information, (ii)

custom data fields, (iii) contact information, (iv) attachments, (v) correspondence, (vi) consent information, (vii) biospecimens and (vii) clinical data. The Ark is also capable of managing complex subject relationships in the Ark-pedigree module (see Section 3.10.8). Subject relationship management is further discussed in Chapter 4. In addition to the Ark-Subject module's data management approaches, it also tracks individual data elements. Researchers can view the history of changes to their subject data elements.

Demographic information

The demographic information section is considered the basic entry point of a study participant in the Ark. The demographic information section records the Subject UID and other basic demographic information related to a subject (e.g., title, first name, last name and date of birth). Based on the Subject UID, the system records other information for the subject in context.

Custom data

In each study, there is information that is specific to a single study dataset. The flexibility of data recording is increased by the use of preconfigured custom fields.

Contact information

Maintaining accurate contact details for subjects is an important aspect of keeping close communication with study participants. The contact section records the subject's postal addresses, email address and telephone number along with other contact details (e.g., secondary address and mobile telephone number).

Attachments

A study dataset can include electronic documents related to its study participants. These documents may be saved as Portable Document Format or other types of electronic formats (Extensible Markup Language, Portable Network Graphics or Microsoft Word. Therefore, the attachment section provides a subject-based document repository that is available for download when required.

Correspondence

Participants are sometimes updated by correspondence from researchers. The correspondence section includes details of communications sent and received from the subjects. The status of emails, phone calls, and mailed letters between researchers and subjects is included in the correspondence section.

Consent information

The consent for, and status of, study components (see Section 3.10.1) for a subject is recorded with the relevant date. The consent implementation is capable of tracking the consent history of a subject with its implications for data usage.

Biospecimens

The biospecimen section records the history of the biospecimens received and associated meta information. Biospecimens are linked to their physical existence, which is discussed in the Ark-LIMS module (see Section 3.10.4).

Clinical data

The clinical data records the subject's questionnaire responses for the study. Selecting a questionnaire shows all the questionnaire answers collected sequentially.

Questionnaires and result sets are discussed in the Ark-Dataset module (see Section 3.10.3).

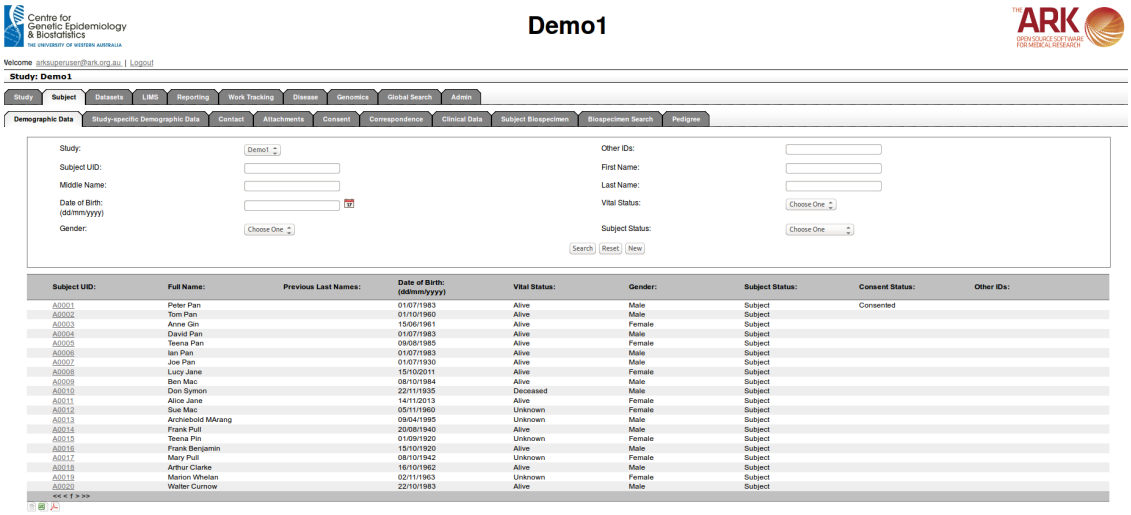


Figure 3.6: The Ark subjects reside in a study list with their unique Subject UIDs.

3.10.3 Ark-Dataset module

Questionnaires are common tools to collect data on study participants. For the Ark, it is important to organise the questions into a logical order, share questions among multiple questionnaires and retrieve the questionnaire responses in a user-friendly manner. The Ark-Dataset module is designed to satisfy these requirements by providing the ability to define a data dictionary to store data for individual questions and arrange these questions into logical groups. Predefined questions in the data dictionary are assigned to categories in the dataset definition phase. Here, researchers can select a category as a questionnaire topic and assign child categories as the sub-topics. The researcher can then assign individual questions defined in the data dictionary to each category. These datasets are a functional representation of a complete subject questionnaire in the Ark. Individual uploaders are available to inject multiple data dictionary items, categories and even a pre-configured dataset

based on existing questionnaire elements. When the design of a dataset is finalised, a researcher can flag it as published and it can then be used to capture responses from subjects.

Researchers can choose a published questionnaire dataset and populate answers to it based on the subject in context. This approach has satisfied the ease of recording the answers to the published questionnaires and later access to the answers of these questionnaires. Based on the common grounds of the questionnaire data management, the Ark has addressed all the required functionalities and is able to handle large questionnaire datasets with less user interference.

Figure 3.7: The Ark dataset definition screen with sample questionnaire. This questionnaire includes available data dictionary fields and categories to define a dataset. From the available categories and dictionary fields the researcher can form a unique category and dictionary field dataset.

3.10.4 Ark-LIMS module

A typical biomedical laboratory setup stores many types of samples (e.g., blood, saliva and DNA) obtained from study participants. These samples are kept in special storage locations to preserve their physical characteristics. Portions are

extracted from the samples for testing and results need to be stored. To keep the samples in secure storage, accurately monitor the sample status, and reliably manage the laboratory environment, it is crucial to adopt an efficient laboratory information management system.

The Ark-LIMS module is designed to manage the biospecimens and record their physical allocation in inventory. Ark-LIMS users can monitor the most recent state of a biospecimen and its current location. The Ark-LIMS module captures the initial state of biospecimens along with information about the biocollection event to which it belongs. Biocollection is an event declared in the Ark-LIMS module. Biocollections are associated with both the Subject UID and Collection UID to uniquely identify the event (see Section 3.10.2), name and collection date. For additional biocollection details, users can define custom fields that operate similarly to the subject's custom fields (See Section 3.10.2). Biospecimens are the actual samples collected from the study participants. According to the Ark-LIMS workflow, biospecimens have to belongs to a specific biocollection that has previously been declared. In the biospecimen initialisation state, the Ark-LIMS module records the specimen type, quantity, additives and concentration. A transaction log is maintained for each biospecimen event including initialisation.

To manage a biospecimen's lifecycle transactions, the Ark-LIMS module includes (i) cloning, (ii) processing and (iii) aliquot options.

Biospecimen cloning

The cloning option replicates the selected biospecimen with its existing properties and initiates a new biospecimen. This option to creates multiple biospecimens with similar properties with a single button click. Laboratory data entry operators can save time by using this option without.

Biospecimen processing

Processing makes a new biospecimen that has a different amount compared with the amount extract from the parent specimen. Therefore, the user needs to specify the amount extracted from the parent biospecimen and the new biospecimen's initial quantity in this scenario.

Biospecimen aliquot

Aliquoting always create a new biospecimen with an initial quantity equal to the amount extracted from the parent biospecimen. Here, the user only needs to specify the initial amount extracted from the parent biospecimen.

Barcode labels represent the identity details of a biospecimen in a machine-readable format. It is common practice in laboratories to print barcode labels and attach them to biospecimen containers. The Ark-LIMS module can register the available barcode printers and print the barcode label for a selected biospecimen. In addition to printing, the Ark-LIMS module stores the barcode image against the biospecimen to allow verification of the barcode image and attached label using a barcode scanner. The Ark-LIMS module is capable of bulk uploading biocollections, biospecimens and allocation details.

The Ark-LIMS module can replicate the physical lab setting up to site, freezer, rack and box levels. Researchers can make allocations to specify the column number and row number in a box. The Ark-LIMS module uses a tree representation to let researchers visualise the locations. Researchers can see individual boxes with their available spaces and stored specimens. In the biospecimens option, a researcher can point to the storage allocation by graphically traversing the inventory and pinpointing the storage location. When allocation is completed, the specimen is highlighted in the inventory location and a barcode image can be attached.

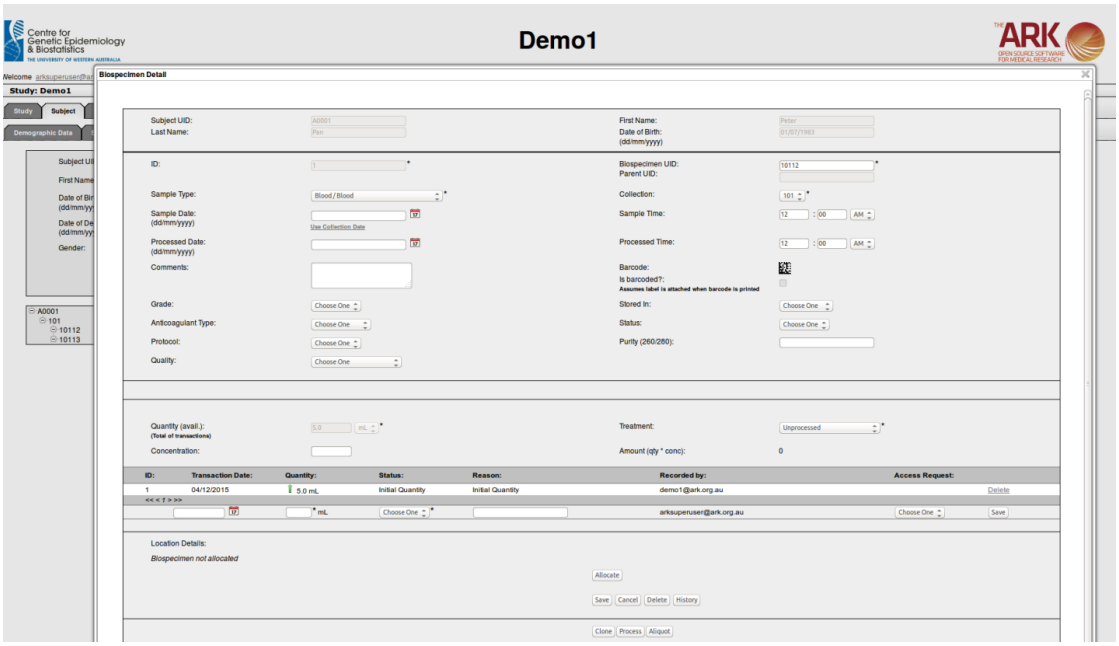


Figure 3.8: Example biospecimen details of a subject in context including the sample's meta information and transactions.

3.10.5 Ark-Reporting module

The Ark-Reporting module was designed to provide summary results based on custom search criteria and output reports as portable document format or comma-separated values files. Reporting is a crucial element to produce summary statistics of the study datasets. There, researchers can view data summaries based on a predefined set of search elements.

Study summary report

The study summary report provides an overview of the subjects in a study in context. The study summary specifies the total number of subjects in the study, the total number of subjects categorised into a status group (e.g., subject, prospect or withdrawal), subject consent status (e.g., consented or not consented) and a total number of subjects consented into study components.

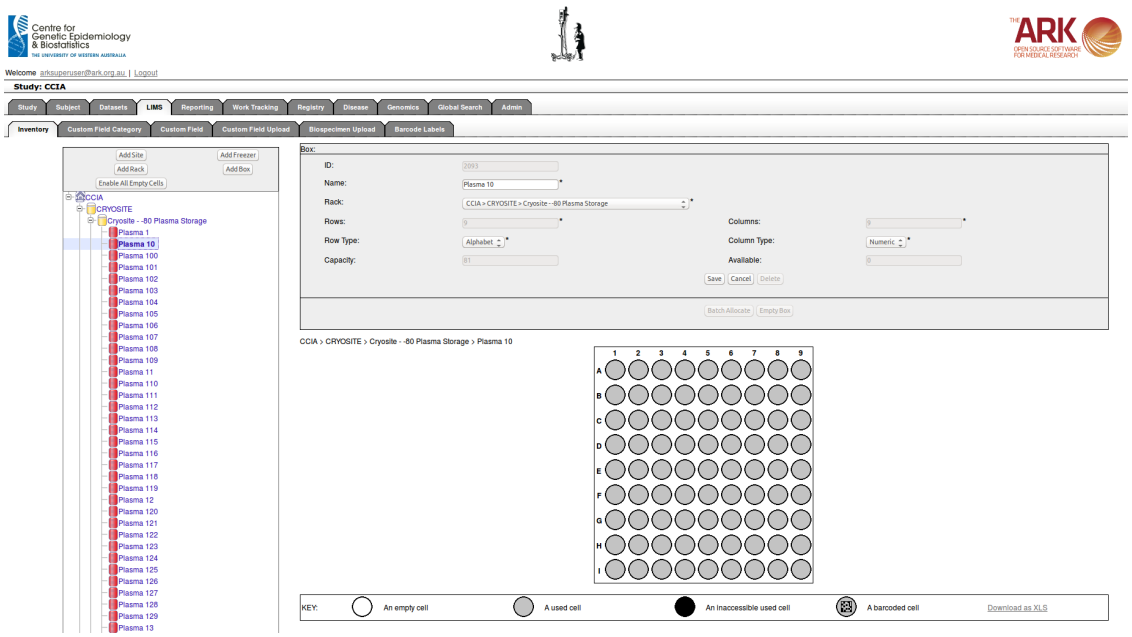


Figure 3.9: Sample LIMS Inventory. Given screen capture represents a Plasma 10 box reside in Cryosite - 80 Plasma Storage rack inside CRYOSITE freezer in CCIA site.

Study level consent report

The study level consent report allows researchers to search on subjects and their consent information based on Subject UID, subject status, consent status and consent date. The report generates a summary of consent events based on the search options for each subject.

Researcher costs report

The researcher costs report provides information on the cost for a specified period by billable item types (see Section 3.10.7). This report provides the information as a tax invoice.

The main advantage of the Ark-Reporting module is its ability to customise a report's output. Rather than using a fixed template for report output, the Ark-Reporting module has been designed to use an individual template per report.

According to study specifications, developers can implement customised report templates identity.

3.10.6 Ark-Data Extraction module

An important function of the Ark is the extraction of subsets of the data for analysis. Flexibility is required regarding the fields extracted and filters applied. The Ark-Data Extraction module was designed to meet researchers' data export requirements. The Ark-Data Extraction module is capable of assigning export scenarios for selected data fields based on logical and relational operators. Once an extraction is defined, the researcher can execute the export as a background job and download the results in a comma-separated values format.

To export data, the researcher needs to define the fields to initiate an extraction. These include demographic fields, subject custom fields, clinical data study components and consent fields from the Ark-subject module; and biocollection fields, biospecimen fields, biocollection custom fields and biospecimens custom fields from the Ark-LIMS module. The researcher can then assign interim logic to declare the field data according to the logical operators provided in the system. Using saved extraction criteria, a researcher can execute a batch job for multiple datasets that reside in the Ark modules. The same extraction criteria can be run at any time in the life of the study, which is useful because study data accumulates and changes over time. Completed extraction criteria contain multiple download files equal to the number of modules defined per criterion.

To further elaborate the Ark-Data Extraction module's process, if a researcher would like to extract the set of male subjects who were born after 31-12-1975 from a study and send a reminder email to participate in a survey. To extract this dataset to populate the automated email application, the following steps would be followed:

Step 1: Select Title, First Name, Last Name, Sex, DOB (Date of Birth), primary email address and Vital Status from the Demographics fields.

Step 2: Create filters including Sex equal to ‘Male’, DOB greater than ‘31-12-1975’ and Vital Status equals to ‘Alive’ and save.

Step 3: Save the data extraction criteria and return to the extraction search list.

Step 4: Click ‘Run Batch Query’ to extract the dataset.

Step 5: When the extraction has completed, click on the ‘Download’ button to open the search results dialogue box. Then click the ‘Download’ button to download the SUBJECTDEMOGRAPHICS.csv file, which contains the extracted dataset.

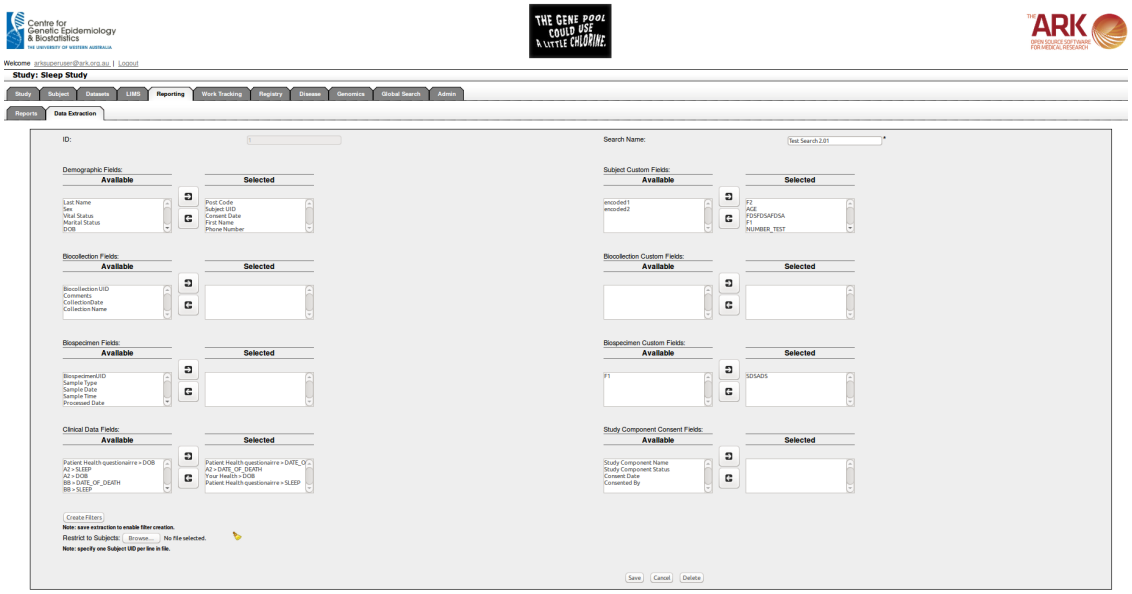


Figure 3.10: Set of data fields select from each module to declare an extraction scenario.

3.10.7 Ark-Work Tracking module

Studies often incur operational costs, especially registries that perform recruitment and follow-up work on behalf of researchers. The details of this work need to

be tracked and , in some cases, billed. To track work activities for invoicing and reporting purposes, the work tracking module was built. The work tracking module consists of the following sub-modules: (i) researcher, (ii) billable item type, (iii) work request and (iv) billable item.

Researcher

The researcher tab allows the recording and tracking of details of collaborators and users of the study as a resource. The Ark is capable of capturing individual researcher details and their accounting credentials.

Billable item type

Billable item types represent the cost components of research. A billable item type is an event that is associated with the cost of a service or good related to the research work(e.g., mail out or phone call). These costs are declared as billable item types with their name, description, unit price and quantity per unit.

Work request

Work requests are made by researchers to specify a task to be completed within a certain period for a certain number of billable item types. This request includes the requesting researcher's name and the status of the event.

Billable Item

Billable items are the physical representation of billable item type assigned to a work request. There can be multiple billable items per work request assigned to a researcher. When the quantity of the billable item is specified, The Ark auto calculate the total cost of the billable item including taxes. There is an option to declare an invoiceable billable item for the accounting purposes.

As an example for a work tracking approach, a researcher has requested a promotion campaign for a survey. The promotion campaign includes mailing and phone call reminders to the prospective study participants. A work request is created for the study survey promotion and billable items are created for the mailed letters and phone calls.

For every billable item, a tax percentage is inherited from its billable item type. The amount of tax is calculated automatically and added to the net total of the billable item. Increasing the count of the mail out billable item adjusts the total amount automatically according to the number of letters sent. The billable item for a phone call assumes a fixed cost per minutes. After completing the survey, the total promotional survey campaign cost can be summarised by a set of reports designed in the reporting module.

The Ark-Reporting module (see Section 3.10.5) includes the following reports associated with the work tracking module: (i) the researcher cost report is an invoice of total costs associated with an individual researcher, (ii) the detail cost report is associated with work requests over a given period and (iii) the study cost report represents the total costs of work requests made by the researchers for a given period.

3.10.8 The Ark-Family Data Management

Family relationship information is often managed as part of study and subject data. Most of the other research study data management systems lack the capability to manage the family datasets but The Ark has addressed this limitation by implementing the Ark-Family Data Management module. This module provides a simple web interface that declares family structure by parental relationships. This enables the Ark to dynamically discover the relatives for a subject in context, visualise the pedigree structures, and add import and export capabilities to family datasets. This contribution is discussed in Chapter 4.

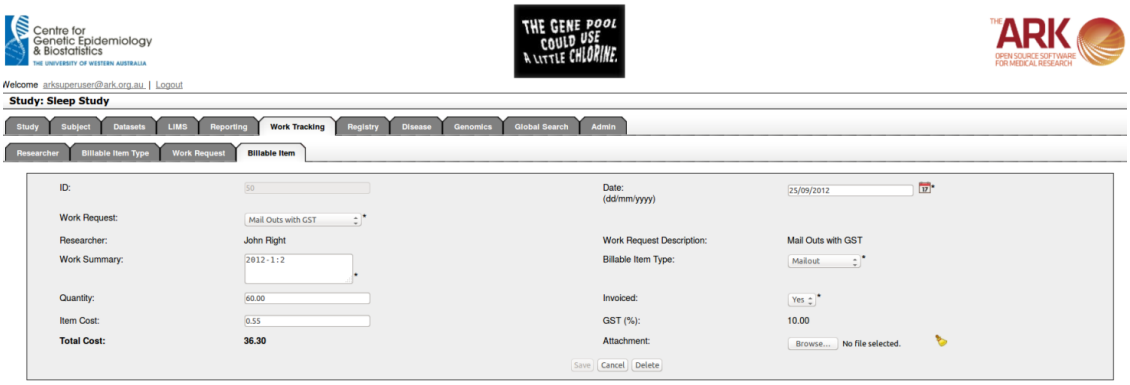


Figure 3.11: Sample billable item: This is a mail out billable item related to the Mail Outs with GST work request assigned to the researcher John Right. The researcher has requested 60 mail outs at a cost of $0.55 each. The billable item type inherits a 10% GST. Therefore, the total cost of this billable item is $36.30. The billable item has been selected as invoiced, and this would bill against the researcher (John Right).

3.10.9 Modules in progress

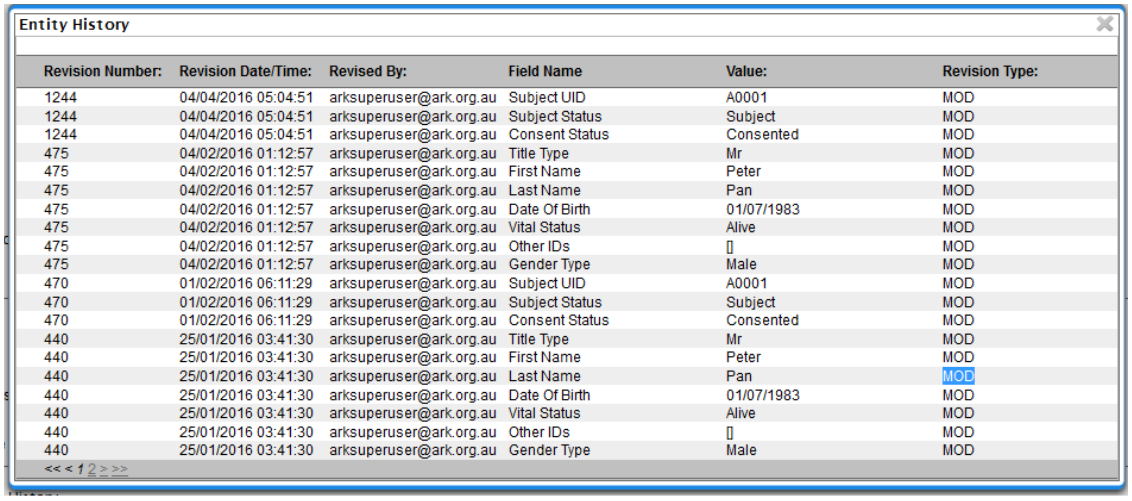
The Ark project is a dynamically growing project. Future modules that will be added to the Ark include (i) Ark-Audit module, (ii) Ark-Disease module and (iii) Ark-Calendar module. These modules are currently in their development and test phases and soon will be available for the real-time study data environments. The Ark-Audit module will allow monitoring of changes in the Ark's data elements. The Ark-Disease module will map disease genes. The Ark-Calendar module will manage the survey participant booking used for meetings and sample collection. Details about the Ark-Audit module are presented below.

3.10.10 Audit module

For a long time, the Ark was unable to keep audit logs for its data. In 2015, developers began the implementation of the Ark-Audit module. Keeping records of changes limits the possibility of losing important information. As an example, subjects' contact details frequently change over time. Keeping a clear record of

changes to contact details can help track environmental changes over time. In the case of an error or mistake, important information is not erased and can be easily reverted to the previous data state.

This concept was supported by the existing hibernate audit management plug-ins. A separate database schema called Audit is introduced to the Ark database. This schema contains audit tables for the matching data table in the Ark's database schemas, and an audit flag enables status in the selected plain old Java object classes to be mapped to the tables. When the audit flag is enabled for a table, every time a create, update or delete event is executed, the event is recorded in the audit tables. The Ark-Audit module's user screen has a history button and that will pop-up history logs for the data record. The Ark-Audit module screen shows who created and modified this record, and the time of these actions. Currently, audits have been implemented only for the Ark-Study and Ark-Subject modules; this is expected to be extended to the Ark-LIMS, Ark-Dataset and Ark-Work Tracking modules.


Entity History

Revision Number:	Revision Date/Time:	Revised By:	Field Name	Value:	Revision Type:
1244	04/04/2016 05:04:51	arksuperuser@ark.org.au	Subject UID	A0001	MOD
1244	04/04/2016 05:04:51	arksuperuser@ark.org.au	Subject Status	Subject	MOD
1244	04/04/2016 05:04:51	arksuperuser@ark.org.au	Consent Status	Consented	MOD
475	04/02/2016 01:12:57	arksuperuser@ark.org.au	Title Type	Mr	MOD
475	04/02/2016 01:12:57	arksuperuser@ark.org.au	First Name	Peter	MOD
475	04/02/2016 01:12:57	arksuperuser@ark.org.au	Last Name	Pan	MOD
475	04/02/2016 01:12:57	arksuperuser@ark.org.au	Date Of Birth	01/07/1983	MOD
475	04/02/2016 01:12:57	arksuperuser@ark.org.au	Vital Status	Alive	MOD
475	04/02/2016 01:12:57	arksuperuser@ark.org.au	Other IDs	[]	MOD
475	04/02/2016 01:12:57	arksuperuser@ark.org.au	Gender Type	Male	MOD
470	01/02/2016 06:11:29	arksuperuser@ark.org.au	Subject UID	A0001	MOD
470	01/02/2016 06:11:29	arksuperuser@ark.org.au	Subject Status	Subject	MOD
470	01/02/2016 06:11:29	arksuperuser@ark.org.au	Consent Status	Consented	MOD
440	25/01/2016 03:41:30	arksuperuser@ark.org.au	Title Type	Mr	MOD
440	25/01/2016 03:41:30	arksuperuser@ark.org.au	First Name	Peter	MOD
440	25/01/2016 03:41:30	arksuperuser@ark.org.au	Last Name	Pan	MOD
440	25/01/2016 03:41:30	arksuperuser@ark.org.au	Date Of Birth	01/07/1983	MOD
440	25/01/2016 03:41:30	arksuperuser@ark.org.au	Vital Status	Alive	MOD
440	25/01/2016 03:41:30	arksuperuser@ark.org.au	Other IDs	[]	MOD
440	25/01/2016 03:41:30	arksuperuser@ark.org.au	Gender Type	Male	MOD

<< < 1 2 > >>

Figure 3.12: Audit trail of a subject and its properties with changes over time.

3.11 Use cases

The Ark is a production-ready system that has been adopted by several large-scale studies in Australia. Notable examples of the Ark's user base will now be discussed

3.11.1 Lifepool Project

The Lifepool Project[18] was initiated by the National Breast Cancer Foundation and co-founded by Cancer Australia. This project aims to collect medical and pathology data from 50,000 women who live in the state of Victoria, Australia. The project currently comprises over 52,000 participants who have given consent for the collection of de-identified medical information. Data collected so far includes the study participant's demographic information as well as their phenotypic information. Study dataset includes the subject demographic information with personal details, contact information, consents and correspondence details. In addition to the demographic data, Lifepool stores information about the biospecimens collected from study participants and digital mammographic images from screened participants. As large-scale data collection is the primary objective of the Lifepool study, the Ark is an excellent fit for the project's data management requirements.

The Lifepool data collection also includes data collected using machine-readable printed questionnaires that are initially scanned and stored in a separate database. In addition to the printed questionnaires, Lifepool uses a third-party online questionnaire system to interact with study participants. After preprocessing, the datasets are imported to the Ark using its data bulk uploading facility. Currently, the Ark manages most of Lifepool's study participant information using the Ark-Study, Ark-Subject and Ark-LIMS modules.

Recently, the Ark has extended its ability to support the Lifepool study by using external the subject attachment storage from the database tables. External subject attachment has been encouraged by storing all of the scanned questionnaire

[18] `www.lifepool.org.au`

answer forms collected (nearly 50 kb per file) and the set of mammographic images received from the Lifepool participants. External storage helps to bear the database overloading from binary files. At the moment an ongoing set of developments in Dataset module focus on support the questionnaire date set in the Lifepool project, and there is a pending requirement to enhance the Ark-LIMS module to support Lifepool biobanking aspects.

The Lifepool project team interacts daily with the Ark, and there are five researchers working on the system. They are involved in data entry, data validation and bulk uploading. Based on the responses received from the study participants (written questionnaires, phone call interviews, etc.) make changes to the Ark datasets.

3.11.2 Western Australian DNA bank

The Western Australian DNA Bank[19] was originally started by the Centre for Genetic Origins of Health and Diseases at the University of Western Australia to store large sets of biospecimens collected from collaborative research partners including hospitals and research groups. The Western Australian DNA Bank aims to help researchers by preserving samples (e.g., blood, saliva and serum) in a secure storage location and managing the meta-data related to the specimens (e.g., demographics, available quantities and transactions).

The Western Australian DNA Bank is supported by the National Health and Medical Research Council of Australia and currently holds more than 100,000 samples collected from more than 30,000 participants. The Ark was selected to manage the data related to specimens stored in remote facilities due to its capability to capture the subject demographics and biospecimens in an integrated web-based manner. The first versions of the Ark-LIMS module were specifically designed to manage the existing biobank datasets. The Ark helps to keep an up-to-date track

[19] `http://www.gohad.uwa.edu.au/enabling-resources/biobanking`

record of study specimens for the geographically widespread study owners.

Notable studies of the Western Australian DNA bank include The Busselton Diabetes study dataset run by Prof Tim Davis, Dr Wendy Davis and Dr Kirsten Peters [164]. This dataset consists of around 400 study participants with cases and controls. The study has collected demographic information on the participants, various test outcomes including subject physical measurements, cardiovascular assessments and DNA sample specimens. Another study accommodated in the Western Australian DNA bank is the Raine Study birth cohort run by Prof Ian Puddy [165]. The Raine Study is a 20-year medical cohort study started in 1989 motivated by the idea of understanding the causality of nature and origin of birth to the diseases. The study recruited 2,900 pregnant women from 1989 to 1991 and monitored them during their pregnancies. The study then monitored the children each year for the next twenty years. The data collected includes subject demographics, questionnaires and biospecimens.

3.11.3 The Australian Inherited Retinal Disease Register and DNA Bank

The Australian Inherited Retinal Disease Register and DNA Bank[20] was started in 1984 and maintained by the Department of Medical Technology and Physics at Charles Gairdner Hospital, Western Australia [166]. The purpose of this project was to maintain a family registry of persons affected by retinal diseases. Datasets include demographic information and results obtained from the electrophysiological, psychophysical and ophthalmological experiments. Currently, this study has 6,700 subjects who are affected by the retinal disease. DNA samples have been extracted from the 5,080 subjects and have been stored for future research.

The Ark's ability to capture demographic and family information prompted the project to use the Ark as its data management platform. The Ark-Subject module

[20] `http://www.scgh.health.wa.gov.au/Research/InheritedRetinal.html`

is used to manage subject demographic information including personal information, contact details, consent details and family history. The Ark-LIMS module has been chosen to record details of the DNA samples. The researchers involved in this study are also contributing to the development of the Ark.

As mentioned in Section 3.11, the Lion's Eye Institute has adopted the Ark as its biomedical data management solution. Even though the RedCap system is available for biomedical data management, the investigators were interested in the Ark's specialised features to manage the study, subject and biospecimen datasets. The Ark-LIMS module is unique to the Ark, and its features have enabled accurate biospecimen data management. The investigators contribute to the Ark with developer resources and lead the work on developing the Ark-Disease and Ark-Audit modules as generic functionality that is critical to their data management process.

3.11.4 The Melbourne Atopy Cohort Study (MACS)

MACS[21] was established from 1991 to 1994 and included 620 children from families that have a history of allergy. The children's demographic information, biospecimens detail, environmental exposures and family history was recorded every four weeks over the first two years, annually for the next five years, and then at 12 years and 18 years. Before choosing the Ark as the data management system, MACS researchers used handwritten paper documents, Excel files, and Access databases to store data. MACS will soon commence a new phase of follow-up and the investigators are looking to manage the data better. The Ark will manage the new phase of data collection and data from the previous phases will be uploaded to the Ark over time to form a unified management system.

Choosing the Ark as the preferred data management solution was prompted by the Ark's track record of managing Lifepool, the Western Australian DNA bank and the Retinal Disease Register. In the first phase of data migration, the MACS

[21] `http://www.car-cre.org.au/research/projects/macs.php`

study investigators are expecting to upload the subject demographic and contact information. Next, they are expecting to transfer the phenotypic datasets and historical data from existing data repositories.

The MACS study has driven the development of the Ark-Calendar module. The new phase of the MACS study has established pre-scheduled participant consultation across multiple sites. Therefore, a separate web application has been developed to make appointments for the study participants. The Ark-Calendar module manages the consultation sessions. When a session is established in the Ark-Calendar module, that information will be communicated to the appointment booking application to notify available time periods to participants.

3.12 Security

The sensitive nature of biomedical data leads to the utmost need for privacy. Biomedical data security includes authentication (login) and authorisation (data access control) that is fine-grained enough to allow researchers to only see and manipulate the parts of the data that are necessary for them to complete their work.

3.12.1 User authentication

An important security principal is authentication of users who requests access to a software system. The most common access authentication method is verifying the user credentials by a username and password. The user name must be a unique constraint to the system and also need to simple to remember by the user. Email addresses are common types of user names because they are unique and easy to remember. A strong password reduces the burden of unauthorised access to the software system. Passwords should be kept in a strongly encrypted format to further strengthen the user authorisation mechanism. Specialised security protocols have

been developed to maintain the user security credentials.

LDAP is a well-known directory based authentication protocol used to secure access to web applications [167] and other types of software applications. LDAP helps to manage user information over distributed directory services in internet protocol supported networks. Today, LDAP is considered as default authentication standard for application security. LDAP is not only used for application authentication but can also be used as an organisational security hierarchy to manage network utilities to application security. The Ark can easily integrate with existing LDAP servers and map existing LDAP users to users stored within the Ark. When a researcher logs into the Ark, the system first authenticates the user credentials via the LDAP protocol.

3.12.2 User authorisation

The Ark's users are assigned different authority levels that determine system security. These are the system administrator, administrator, data manager and read-only users. The modules assigned to the study are assigned to the administrator, data manager or read-only user role per module. If a study is assigned the Ark-Study, Ark-Subject, Ark-LIMS, Ark-Dataset and Ark-Work Tracking modules, the user roles are assigned per module. For example, the Ark-Subject module has subject administrator, subject data manager and subject read-only user roles. Every access group and associated privileges are clearly declared in the Ark back-end reference tables that control functionality (see Table 2.1) for specific access specifiers. Specifying the access levels helps to reduce the risk of introducing quality control problems over the large datasets by allowing data changes on a minimum needs' basis.

The Ark super user has the administrative privileges for the settings and datasets. The super user role permits creating studies and sub-studies, assigning modules to the studies, creating and modifying users, unconditionally accessing all modules,

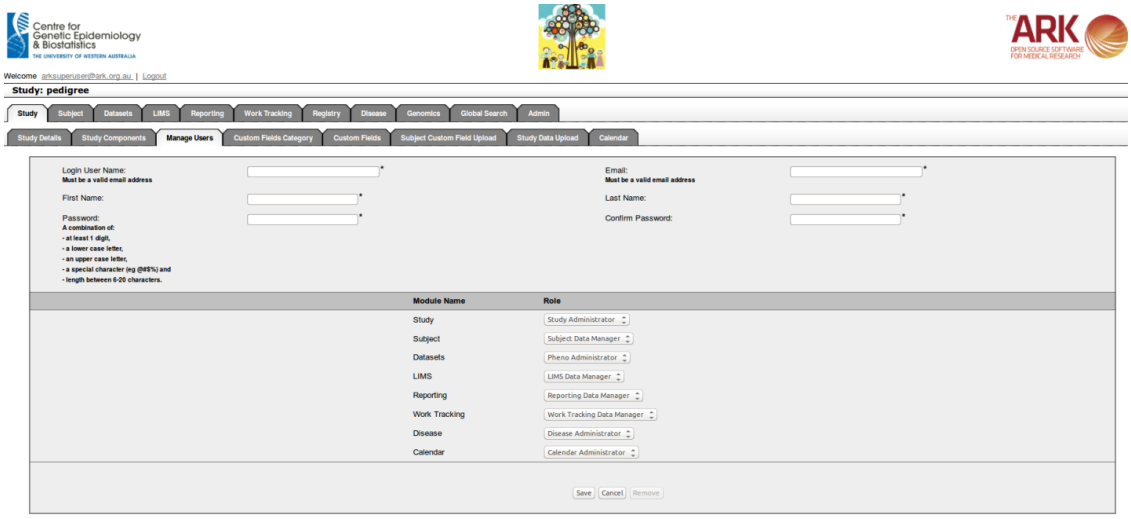


Figure 3.13: The Ark user with the different user roles assigned per module.

Role	View	Create	Modify	Delete
Administrator	Yes	Yes	Yes	Yes
Data Manager	Yes	Yes	Yes	No
Read-Only	Yes	No	No	No

Table 3.1: Summary of the Ark user roles and their privileges over the datasets

enable users to execute processes, creating the Ark-LIMS sites, and unconditionally access and modify any dataset. The module administrators can create, delete and modify data residing in a module, while the data manager can only create and modify the datasets. Read-only users can only view the data in a module and cannot add or change existing datasets.

3.12.3 Other security aspects

The Wicket web application framework is widely used for secure web application development. According to the application specifications, Wicket has authentication and authorisation mechanisms to validate user logins and heir user role privileges.

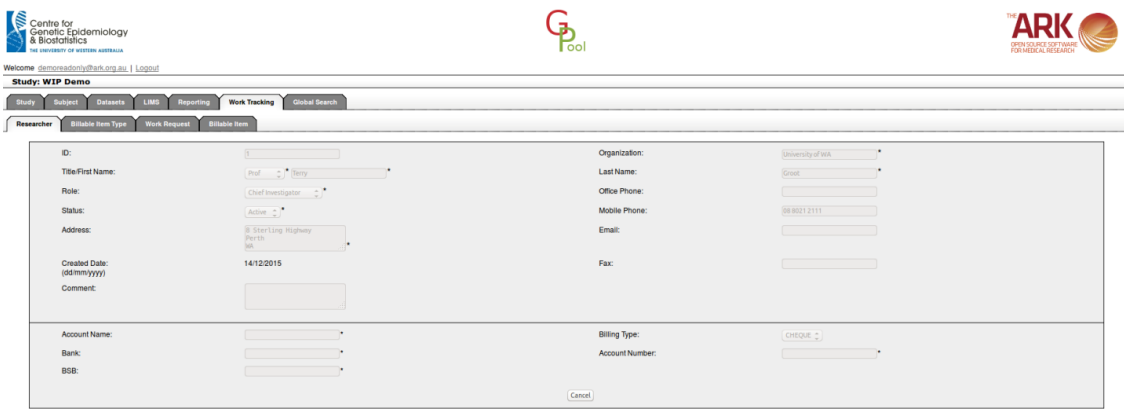


Figure 3.14: The Ark detail form with read-only access permission view where all the user controls on the screen are disabled and the "Save" and "Delete" buttons are hidden.

According to the Wicket application programming interface, Wicket provides an authenticated web session to validate login credentials and user roles. As an example, the secure Wicket application monitors the activities of a web page and has a pre-configured cut-off time to disallow requests from an idling web page pre-configured in the web session timeout. Wicket also has an inbuilt mechanism to map secure web calls with universal resource location encryption to hide details files from public view. The Spring framework has secure plug-ins to map the LDAP database to the Wicket web front end.

Web application hosting best practice is to separate public web interfaces from application data management so that the main web application and supporting services such as databases and LDAP servers are hosted separately. The Ark's web servers are hosted in secure infrastructure and guideline to security specifications are provided for expert system administrators. The network firewalls have been configured to monitor incoming and outgoing network traffic and deny harmful incoming requests to the application hosting network. As an additional precaution, only the required ports are enabled in the hosting server to protect the application servers from intruders. The Ark's hosting servers have purchased digital certificates to verify their identities to the clients and enable secure message passing between

client and servers.

3.13 Similar systems

As discussed in the Introduction, systems that are similar to the Ark have been developed by research groups across the globe. According to the literature review (see Chapter 2, Section 2.3.2), selected set of biomedical data management systems are developed with similar functionalities to the Ark system. Therefore, advantages of The Ark when compared to the other systems are discussed in detail.

The REDCap [31] system is the most cited biomedical data management solution. According to the REDCap website[22], it has been deployed in 99 countries and partnered with 1,832 research institutions. REDCap started as an online survey database for medical researchers and has extended its capabilities over time. REDCap enables researchers to design questionnaires using multiple data types. The philosophy behind REDCap is to see datasets as logically ordered questionnaires and manage multiple questionnaire datasets per study.

As with the Ark, REDCap can host multiple studies per single installation and can initiate different user privileges per study. Even though there are similarities in study management and use of custom data fields, there is a significant difference in REDCap's approach compared with the Ark. The main difference is that the Ark has specialised modules for each data category. For example, the Ark-LIMS module was specifically developed to manage the laboratory datasets including monitoring the specimen transactions and storage allocations. The Ark-Pedigree module is capable of storing, recording and visualising family datasets. It is quite hard to implement a new module in the REDCap system because its non-modular implementation mechanism restricts much of the independent development process. Also, REDCap has a generic data schema implemented in its data layer, and this has limited data mining possibilities to specific data categories.

[22] `http://project-redcap.org/`

Slim-Prim [82] is another web-based biomedical data management system. Based on the directives of the Health Science Center at the University of Tennessee, the system operates with a cost recovery basis. The application is limited to single study data management and the Oracle 11g database server backs data storage. The system is capable of importing different data formats including text files and Microsoft Excel spreadsheets with a set of pre-configured mapping instructions. The mapping process is supported by popular medical terminology data semantics (e.g., ICD and SNOMED CT). Slim-Prim also has the capability of generating complex queries and outputting results in statistical analysis formats (e.g., R, SAS and SPSS).

Slim-Prim is a single study-oriented system. Therefore, it does not have the flexibility to manage multiple studies owned by different research groups. The Ark consists of specialised data management modules, but Slim-Prim does not have such capabilities. Slim-Prim is based on a predefined set of data definitions and from which data elements are mapped. An important functional feature of Slim-Prim is its ability to export data files in statistical application file formats. The Ark currently only exports data in comma-separated values formats, but this can be further enhanced to different formats in the future.

Translational Data Marts [83]is a system developed to manage phenotypic and genotypic datasets generated from existing biomedical studies. Translational Data Marts is a data warehouse solution to integrate various data sources via workflows rather than as a real-time data management system. Translational Data Marts has the capability to manage multiple user accounts with different access privileges. The system has the predefined workflows to manage the demographics, questionnaire, biospecimens, clinical trial results and genotypic results data.

The Translational Data Marts and the Ark have many similarities. The main difference is the semi-automated workflows declared to the data operations in the Translational Data Marts, where the Ark operated based on study-subject context data management mechanism. The Ark is capable of specialised data management

approaches to pedigree and work tracking datasets. Multiple study data management is a feature of the Ark, and the Ark-Genomic module manages genomic datasets.

OpenClinica [81] is a hospital-based biomedical data management repository. It has a clear focus on patient demographic data management for multiple studies. OpenClinica manages multiple studies per single installation with user role privileges. OpenClinica does not have specific mechanisms to manage questionnaire or biospecimen datasets. It has its clinical data record mechanism with the option to record treatment and test scenarios (e.g., magnetic resonance images and laboratory results).

When checking the availability of these systems, Slim-Prim and Translational Data Marts are closed source systems. These systems are limited only to the users and developers who already have access rights from the owners. REDCap and OpenClinica source codes are in public source code repositories. Table 3.2 compares the features of the Ark and other biomedical data management systems.

3.14 The Ark genomic data management and analysis

The genome-wide association study era has led to a significant demand for genomic data management systems. There are two significant challenges for software developers producing these systems. First, genomic datasets exceed the capacity of traditional relational database management systems, and second, the required high-performance computational power to process and analyse these datasets exceeds the processing power of a typical desktop computer. The Ark's users have some genomic datasets related to existing phenotypic datasets. So, there was a need to implement a genomics data management system based on The Ark. The SPARK

Features	The Ark	REDCap	Slim-prim	Translational Data Marts	Open Clinica
Study information management	Yes	Yes	No	No	Yes
Subject demographic information management	Yes	Yes	Yes	Yes	Yes
Subject pedigree information management	Yes	No	No	No	No
Phenotypic information management	Yes	Yes	No	Yes	No
Laboratory information management	Yes	No	Yes	No	No
Work tracking	Yes	No	No	No	No
Reporting	Yes	Yes	No	No	Yes
Data import/export capability	Yes	Yes	Yes	Yes	Yes
Data validation	Yes	Yes	Yes	Yes	Yes
Data de-identification	No	Yes	No	No	No
Role-based data security	Yes	Yes	Yes	Yes	Yes
Web interface	Yes	Yes	Yes	Yes	Yes
Configurable data fields	Yes	Yes	No	No	No
Bulk data uploading	Yes	Yes	No	No	No
Audit tracking	Yes	Yes	No	No	Yes
Pedigree Visualization	Yes	No	No	No	No
Genomic data management	Yes	No	No	No	No

Table 3.2: Comparison of the Ark with other biomedical data management systems.

project addresses these challenges and is built upon the existing capabilities of the Ark.

The SPARK project was designed to build upon and extend the Ark framework and its existing data management approach. The Ark-Genomics module operates as a common web interface for a set of micro-services generated by SPARK web nodes. Importantly, the Ark is a data management platform rather than analytical pipeline. The SPARK web nodes were developed as independent high-performance big data management and high-dimensional analysis extensions.

The SPARK project has adopted cutting edge software architectural designs to fulfil these objectives using the latest open-source software frameworks. Chapters 5 and 6 provide an in-depth discussion of the SPARK system, its design objectives, development goals and architectural design. Evaluations of the system's implementation quality and researcher satisfaction are presented in Chapter 7.

3.15 Future enhancements

The Ark has been developed by contributors across Australia. Compared with other biomedical data management systems, the Ark has fulfilled most of the functionality required to manage a biomedical study dataset. The Ark needs to implement a de-identification mechanism to support the requirements of ethics agreements, and this is especially important when datasets are exported for dissemination or external analysis. The de-identification process consists of selecting a set of data fields and replacing the values in those fields with system-specific encoded values. The decoding process is unique to the system and implemented using an encryption algorithm.

The Ark also needs to implement a web service layer to integrate the Ark with third-party software systems. The Ark's lack of a communication mechanism to integrate with Health Level Seven International or Fast Healthcare Interoperability Resources specifications. These are the preferred communication specifications to

integrate with electronic health records. To integrate the subject research data in the Ark with electronic health records, integration services that support Health Level Seven International and Fast Healthcare Interoperability Resources specifications need to be implemented.

Laboratory information management is important functionality embedded in the Ark. An automated process implementation for specimens is available, but the system lacks a bulk extraction facility. Development of a bulk processing workflow has been requested from many biospecimen owners who use the Ark. These users have requested a wizard-based selection process to mark a set of biospecimens in a study and specify a common extraction volume. A single extraction identifies a shipment and a common amount needs to be deducted from the original specimen volumes. A sample workflow would be where a biospecimen collection identified by collection UID is selected and specimen samples are extracted from the collection to create a shipment of extracted biospecimens. This extraction automatically adjusts the parent specimen volumes according to the amount extracted.

Data de-identification is an important task of biomedical data management due to the ethics requirements and data management best practice. The Ark declares user role privileges for users but does not imply user role privileges to individual data elements. As discussed in Section 3.12.2, the system can prevent user access to the Ark modules and restrict user interaction to data fields with data editing privileges. Some researchers have suggested that it would be more useful to have data field visibility configuration per user group. The data extraction process will be modified to encrypt certain fields to restrict the important information by avoiding visibility to unauthorised users. There will be a data microscope to decrypt these encrypted data fields using the Ark.

The Ark currently restricts its data export only to comma-separated values files. The reporting output is not sufficient to seamlessly interact with popular statistical analysis tools such as R, Stata, SAS and SPSS. Therefore, future enhancements will address this requirement by implementing a multi-format data export facility.

3.16 Conclusion

This chapter has presented the Ark, an open-source web-based biomedical data management tool. It has highlighted the importance of using biomedical data management systems for medical research by historical examples and current literature. This chapter has also highlighted the success of the Ark's data management approach in existing medical research studies. This chapter has discussed the importance of using open-source software frameworks and tools for rapid application development, and how they helped to boost the quality of biomedical data management. Agile project management has proven its suitability for a wide range of application development projects, and the Ark has provided a successful adoption of agile Kanban project management methodology for biomedical data management systems. This chapter has provided a summary of the Ark's capabilities including managing multiple studies and sub-studies in a single database, laboratory information management and questionnaire data management.

This chapter has provided a detailed description of the Ark's architecture and its modular design. The modular architecture has helped the Ark's distributed development environment and its rapid development phase. The Ark's role-based security infrastructure has provided a secure environment for biomedical datasets. It gives system administrators the ability to restrict access levels for users based on their roles. The Ark's capabilities were then compared with other biomedical data management systems. The feature comparison of the Ark with other biomedical data management systems shows the importance of using the Ark for biomedical data management.

According to the literature, neither of the study-based biomedical data management system tried to manage the pedigree datasets. The Ark pedigree module has addressed this challenge in a study-based relational data environment. Chapter 4 will discuss the Ark's approach to pedigree data management.

An open-source, integrated pedigree data management and visualization tool for genetic epidemiology [168].

4

Pedigree Data Management

4.1 Introduction

Chapter 4 presents a novel approach to pedigree data management. The chapter starts with an overview and explanation of the common characteristics of pedigree datasets. Existing software approaches to manage pedigree datasets are then discussed, and an examination of the features of popular pedigree data management systems clearly indicates the importance of pedigree data management for research in an open-source platform.

The Ark pedigree module was developed to have the features available in commercial pedigree data management systems. A significant challenge was matching existing pedigree data management techniques to a research data management en-

vironment because pedigree identity - which is central to a pedigree dataset - does not always appear in research data management systems. The studies reside in a research data management system are identified by a unique identity and a study based identity is assigned to the participants. This chapter discusses a data model to hold family relationships in an existing research-based relational data schema. This enables a novel approach to pedigree data management that eliminates pedigree identity as a key constraint when only parental and twin relationships are stored.

When storing relationships within a pedigree, there has to be a mechanism for extracting relationships for each subject. Rather than storing relationships for each subject, the Ark pedigree module is capable of discovering relationships via an algorithm. The characteristics of the pedigree discovery algorithm, called BloodLine, are discussed in terms of its pseudo code, space complexity and execution time complexities.

Precise pedigree dataset can be discovered for each subject according to the relationships recorded in the system. Therefore, validation has played a crucial part in pedigree data management systems. In non-pedigree identity approaches, it is particularly important to accurately record the relationships between subjects. This ' discusses the Ark pedigree module's implementation of a set of input restrictions for creating relationships via the web interface and the introduction of a graphical validation mechanism to detect and warn of consanguineous relationships.

Pedigree visualisation is a key aspect of pedigree data management. Therefore, the Ark is integrated with an external pedigree rendering engine called Madeline to visualise the relationships discovered for a subject in context using the BloodLine algorithm. The ' concludes with a discussion of the software architectural approach to manage pedigree datasets in a research environment and the technical availability of the work.

4.2 Pedigree data management

I now discuss the nature of pedigree datasets, existing and popular approaches to managing family datasets, and opportunities within open-source research data management systems to further develop pedigree data management.

4.2.1 What is a pedigree dataset?

A pedigree dataset is a special type of data that contains information regarding the relationships between the subjects and subject-related family history. These datasets have been widely used to understand the genetic causes of hereditary diseases. When analysing pedigree datasets, researchers have understood that there is a high possibility of a disease being present among multiple family members. This occurs because family members have shared a common genetic blueprint, and most have been exposed to similar environmental conditions.

Family history has been considered as a critical diagnostic factor when investigating certain clinical diseases (e.g., type II diabetes, coronary heart disease, asthma and cancer) [169], and medical researchers have recognised the importance of storing family history data to use in their analyses. Research carried out on Mendelian diseases has shown the importance of studying the family histories of affected people [47] and genetic testing for selected family members has helped to identify the genes responsible [170].

Pedigree datasets can be grouped in to two categories. The first type is birth-oriented, where relationships are built between the family members based on birth. The family members in these datasets have a guaranteed genetic relationship between them. The second type of pedigree dataset is based on social relationships, particularly marriages . This category can be considered an extension to birth-oriented pedigree datasets. For this category, marriage partners are added to the birth-oriented pedigree members. While birth-oriented pedigrees reflect the biological relationships among family members, marriage-oriented pedigrees reflect the

socio-economic relationship between them.

4.2.2 Existing approaches to the management of pedigree datasets

Progeny[1] is one of the most popular pedigree data management platforms [40]. It is capable of handling pedigree datasets that are based on births and can also incorporate marriage structures. Progeny manages relationships and the display of pedigree diagrams via a user-friendly graphical user interface client, and it provides capabilities for bulk import and export of pedigree datasets. Progeny's data management is bound to the pedigree identity and pedigree members are grouped based on a unique pedigree identity constraint. Progeny has its own database and can be configured as a desktop or web application in a Windows operating system. Progeny is proprietary software, and users have to adhere to the commercial agreements with its creators. In addition, users need to follow the practices and work flows contained in Progeny.

4.2.3 Motivations to study pedigree data management

Pedigree datasets have unique features that separate them from a dataset of unrelated individuals. Pedigree data often focuses on a single subject called the proband, and the recorded relationships between members of a pedigree are based on the degree of relatedness to the proband. Compared with a dataset of unrelated individuals, a pedigree dataset requires more time to establish and more stringent quality control mechanisms to avoid data ambiguities that may occur in complex pedigree structures.

Much of the research attention on pedigree datasets has been directed towards pedigree visualisation. In the mid-1990s, a task force was established by the Professional Issue Committee of the National Society of Genetic Counsellors to standard-

[1] `http://www.progenygenetics.com/`

ise pedigree visualisation [171]. Based on these standardisation guidelines, various research groups have developed tools to visualise complex pedigree datasets using techniques such as the H-tree layout [172, 173, 174], two-dimensional and three-dimensional images [175]. A number of other pedigree visualisation tools have been developed [176, 177, 178, 179] that require different data input formats. Pedigree datasets that are held in research data management systems often have to be manually extracted for visualisation purposes. Based on the shortcomings of existing approaches, the Ark designers were motivated to think about a novel methodology to manage pedigree datasets in a research environment.

4.3 Pedigree data management requirements in a research environment

Several biomedical data management systems have been developed to manage data in a research environment. Specialised software systems exist for management of research datasets, and one example is the Ark, which was introduced in 3. Most of these systems lack the ability to manage pedigree datasets. On the other hand, systems that are able to manage pedigree datasets [47] are unable to support broader research data management requirements. Therefore, many pedigree datasets have been incorporated into research datasets and pedigree information is recorded as properties of an ordinary research dataset.

Importantly, the Ark is open-source software. Unlike commercial products, Ark researchers can use the pedigree data management tool with their research datasets at no cost. The flexibility of the design and free source code enables Ark users to customise the system based on their preferred pedigree management practices. This gives researchers the freedom to use their own pedigree visualisation platform with minimum changes to the system.

The Ark pedigree module was designed after a thorough examination of the properties of a pedigree dataset and existing functionality of research data man-

agement systems. This includes the functions described below in Sections 4.3.1 to 4.3.3.

4.3.1 Declaring and discovering the pedigree relationships per subject in context

The Ark was designed to manage multiple research datasets inside a relational database environment. The pedigree module for the Ark has to be compliant with the existing data model. It needs to establish relationships between existing research subjects and navigate through the subject's relatives via the Ark user interface. It is important that the pedigree module dynamically discovers the relatives for each subject in context based on relationship types. The Ark pedigree module does not rely on family identities because subjects are grouped based on studies to be compliant with The Ark data management policy.

4.3.2 Visualising the pedigree relationships

A number of standalone visualisation tools have been developed for existing pedigree structures [180]. Therefore, the Ark's pedigree module does not need to develop a new pedigree visualisation engine; instead it needs to integrate an existing pedigree visualisation engine. This has to be a widely accepted visualisation tool that can easily integrate with existing system architecture. The pedigree visualisation module has to be enabled for every subject in context and dynamically generate the pedigree diagram for discovered relatives.

4.3.3 Pedigree data import and export functionality

The Ark is capable of bulk uploading datasets. Similarly, the Ark's pedigree module must allow data to be uploaded to the system. The challenge is to produce a template of the Ark's pedigree data upload file that will have broader applications.

The pedigree data needs to be able to be exported as standard pedigree data files and printer-friendly pedigree diagrams.

A single web user interface needs to be designed to manage the requirements listed above in a user-friendly manner. This includes, using the existing user controls of the Ark and re-paintable rendering screens to visualise the pedigree diagrams.

4.4 The Ark's approach to managing pedigree datasets from its pedigree module

The Ark's pedigree module was designed based on the requirements identified in Section 4.3. After analysing existing datasets, it came into attention that any birth-oriented pedigree relationship can be described as either a parental or a twin relationship based on the subject in context. To match the relational data model of the Ark, a normalised table structure to store parental and twin relationships for each subject is introduced by this thesis. Pedigree-specific web controls were then introduced to the Ark web interface to select parent subjects, assign twin status among the siblings, visualise the pedigree structure of the subject in context and configure the pedigree settings. To extract the relatives related to the subject in context by birth, algorithm namely BloodLine has been developed. The BloodLine algorithm automatically discovers the blood relatives of a subject in context and builds a single pedigree structure using parental and twin relationships.

4.4.1 Declare the relationships between subjects in a study

The Ark system manages the subjects for a specific study. Therefore, pedigree members are a sub-set of the Ark's dataset. In an ordinary pedigree data management system, families are bound to a pedigree identity key. Here, each pedigree

member is a subject residing in the study and has a unique subject user identity (UID). Using a pedigree identity to manage the relationships between the subjects in a study dataset diminishes the flexibility of the data management system. The system's capacity to dynamically assign study subjects to multiple pedigrees is reduced, as is the ability to manage subjects without a pedigree identity.

The Ark dynamically discovers the pedigrees based on the parental and twin relationships. This approach has eliminated the requirement to record a pedigree identity for study participants. Using an optional subject demographics field to represent the family identity enables the system to record family information independent to the subject relationships.

The Ark's data management system can easily record the parental relationships for each subject. While every human being has a father and a mother, this parental information can sometimes be unknown or forgotten after long period. Gender and date of birth are commonly recorded demographic fields in research data management. Therefore, system enables researchers to discover the parental relationships based on The Ark subject search criteria.

Twin relationships are categorised as monozygotic or dizygotic. All twins are siblings who share the same parents. By referring to existing parental relationship records, the system can select siblings who are eligible for twin relationships. Identified siblings are then assigned the correct twin type. Therefore, known twin pairs can be declared and discovered when building the pedigree relationships.

Set-up parental relationships

In the Ark pedigree module, it is important to declare accurate parental relationships for the subjects in a study. False parental relationships will violate the BloodLine algorithm. Therefore, the system has gender restrictions in selecting the parental subjects and validates the subjects against the younger generations. In addition, age restrictions are applied to the subjects who declared the date of birth

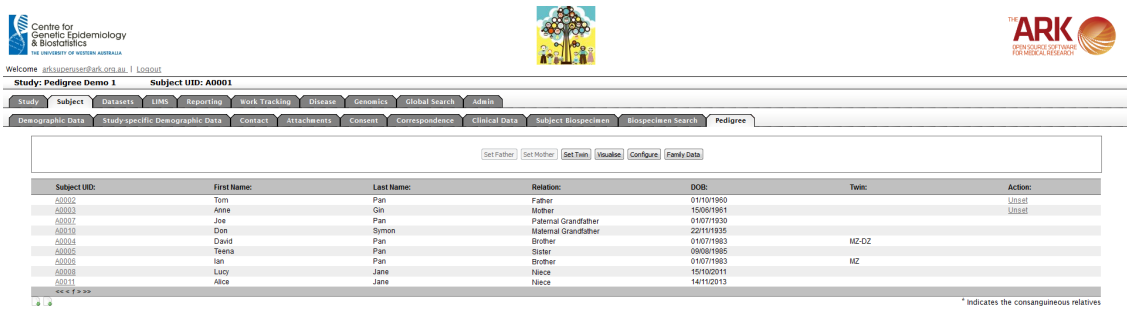


Figure 4.1: The Pedigree module start-up screen with Set Father, Set Mother, Set Twin, Visualisation, Configure and Family Data buttons. In addition, the subject in context's (A0001) blood relatives list is displayed.

to avoid invalid parental relationships. Only select the parental subjects whose date of birth is earlier than the subject in context.

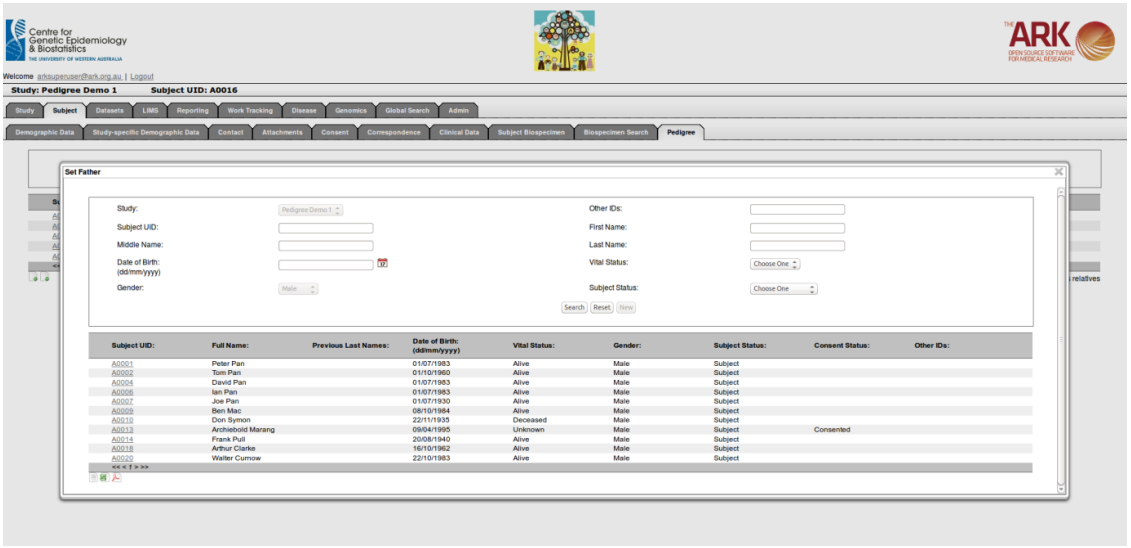


Figure 4.2: Clicking the Set Father button opens the dialog with a list of male subjects in the study. The researcher can click one of the Subject UID links to set the father for the subject in context.

Set-up twin relationships

Twin relationships are the next step in generating the pedigree structure. It is important to accurately record twin relationships among siblings who share the

same parents. The Ark twin relationship set-up was developed based on siblings who share the same parents and ignores the date of birth constraint when data are missing.

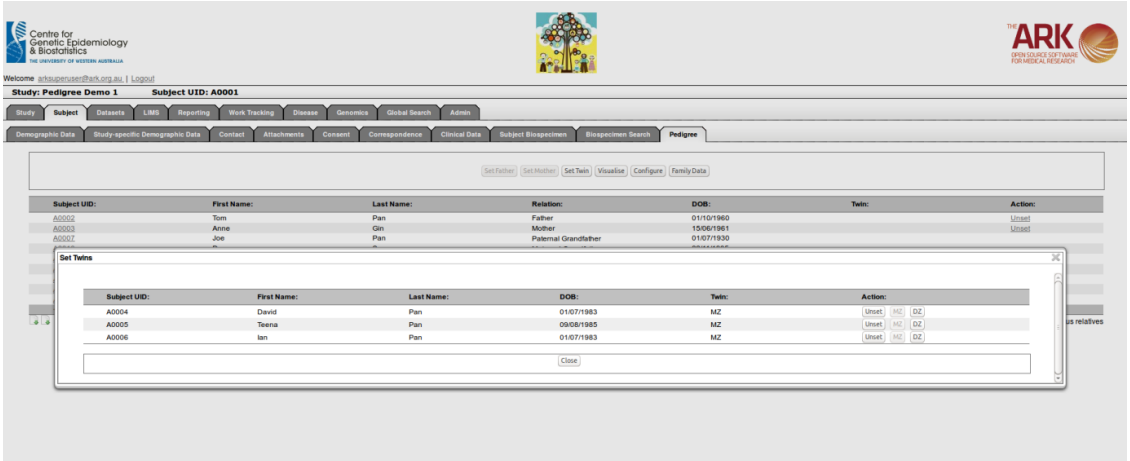


Figure 4.3: Declare the twin relationship between the subject in context and his or her siblings by clicking the MZ or DZ button according the twin type. Existing twin relationships can be discarded by clicking the Unset button.

4.4.2 Visualising the existing pedigree structures

Pedigree visualisation is a key requirement of a pedigree data management system and must be considered when pedigree datasets are incorporated into the overall data management system for a study. Unlike a dedicated pedigree data management system, in a broader research data management system it is necessary to visualise dynamically discovered pedigree datasets that are based on a subject in context. This is an important consideration when selecting an existing tool to dynamically generate a pedigree diagram for each subject in context.

The selected pedigree visualisation tool has to be compliant with the existing standards of the Ark. Therefore, it has to be open-source and able to be easily integrated with the Ark. In addition, the tool has to be able to be customised according to the Ark's requirements and synchronise the image stream on web

application display panels. Security and fail safe mechanisms need to be compliant with the Ark's standards.

Based on the above conditions, the Madeline [181] pedigree rendering engine was selected for pedigree visualisation. Using this tool, the Ark is capable of generating the pedigree diagrams for each subject in context and rendering the image on a web panel. The Ark has simplified this process by allowing researchers to visualise the subject in context's pedigree diagram by clicking on a single button.

4.4.3 Importing and exporting pedigree data with the Ark

The Ark is capable of bulk uploading datasets. Similarly, pedigree datasets can be imported to the system using an Ark-formatted pedigree data file. This approach has helped researchers to bulk upload existing family relationships as parental and twin relationships. The only prerequisite is that the individuals in the relationships have to be existing subjects of the Ark. Based on the format in Figure 4.4 below, researchers can migrate the pedigree datasets to the system as a single batch process.

IndivUID	FathUID	MothUID	TwinStatus	TwinUID
Mandatory	Mandatory	Mandatory	Mandatory	Mandatory
	-	-	-	-
	-(dash) = Missing	-(dash) = Missing	-(dash) = Not Twin M=Monozygotic D=Dizygotic	For M and D TwinStatus, the IndivID of the twin, otherwise - (dash)

Figure 4.4: Downloadable pedigree data import template from The Ark.

Data exporting is an important aspect of data management. Based on the subject in context, researchers can export the blood relatives as a text data file or an image file. This allows researchers to extract text formatted as a pedigree dataset in either the Madeline ped file format or the Ark ped file format. The

intention of this approach is to give researchers the opportunity to re-upload the pedigree data to the stand alone Madeline tool, re-upload to the Ark or make a back-up copy of the subject's blood relative list.

The pedigree visualisation image is important for publications and presentations. Therefore, researchers can download a pedigree image from the Ark for each subject in context. Downloadable images can be either portable document formatted (PDF) or portable network graphics (PNG) files. PDF files are considered more secure and take less disk space because they are compressed. PNG files are easier to edit and are more useful in applications such as graphics editors. In addition, PNG files can be easily transformed to other graphics formats (e.g., JPEG or BMP).

4.4.4 Configuring the pedigree visualisation options and selecting a family identity

Researchers may gain substantial benefit from being able to customise the Ark's pedigree visualisation options. A set of flexible configuration criteria allow researchers to control the information presented in a pedigree diagram. Most importantly, researchers can select a set of data fields to be displayed in the pedigree diagram. For example, affected status is an important identification marker for heritable diseases among the family members, and researchers can choose to display this or other custom fields on the pedigree diagram. In addition, the pedigree module allows researchers to plot date of birth and current age or age at death of the pedigree members.

While pedigree identity has been omitted as a constraint in the pedigree module's design, the pedigree identity in pedigree data management cannot be totally ignored. It is the only foreign key to mark the pedigree's meta information. Therefore, the Ark pedigree module uses one of the subject custom fields as a pedigree identity and records family meta information with the pedigree custom field data. Allowing consanguineous relationships can lead to data entry issues in large

datasets. Consanguineous relationships exist in some pedigree datasets and the Ark pedigree module can be configured to allow or disallow consanguineous relationships among the subjects.

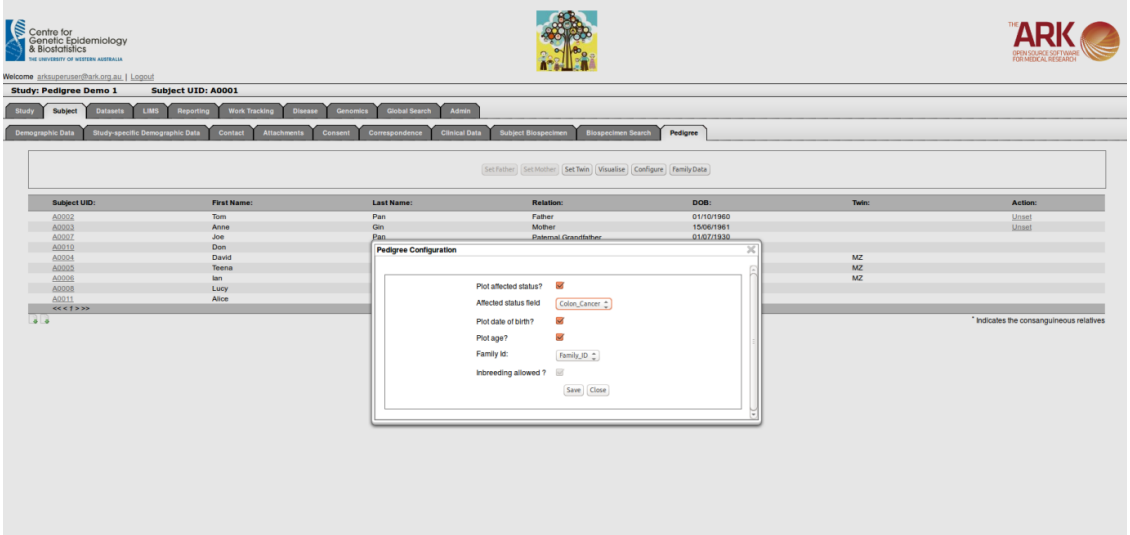


Figure 4.5: The pedigree configuration option enables visualisation options, Family ID selection and whether inbreeding is allowed for the study.

4.4.5 Pedigree meta information management with custom fields

' 3 described the use of the Ark's custom fields to manage subject properties. Similarly, the Ark's, pedigree module uses the pedigree custom fields to manage family meta information. Pedigree meta information consists of any data value that is referenced by the subject's pedigree identity. Therefore, character, number or date-based fields can be represented in the pedigree data section. When the researcher selects a pedigree identity field in the configuration section and enters a pedigree identity value, the system automatically enables the pedigree data section per subject. All of the pedigree custom data fields are then shown with values assigned to them. When a researcher edits a pedigree custom field value, the changes are recorded for all other subjects who share the same pedigree identity.

Managing meta information in the pedigree module enables researchers to maintain consistent data flow in their pedigree datasets. Maintaining a unique Family ID among a set of pedigree members can generate clusters among the blood relatives discovered from the BloodLine algorithm. This helps researchers to fine tune their work by introducing additional flexibility in assigning individuals members to a pedigree. Automatic retrieval of the pedigree's meta information for subjects sharing the same pedigree identity helps researchers to maintain a continuous data flow and provides a mechanism to share information between pedigree members.

Figure 4.6: A pedigree's custom fields assigned to the study will populate inside the dialog and repeat the information for subjects who share the same Family ID.

4.5 Data model

The pedigree module's data schema has to be compliant with the table structure in the Ark and follow the study, subject table relationships. The entity-relationship diagram in Figure 4.7 below demonstrates the Ark's table structure before the introduction of the pedigree table.

The pedigree data schema needs to operate as a generic table schema for any study subject-based data management system. Based on that and the requirements discussed above a general table schema to hold subject pedigree dataset has been

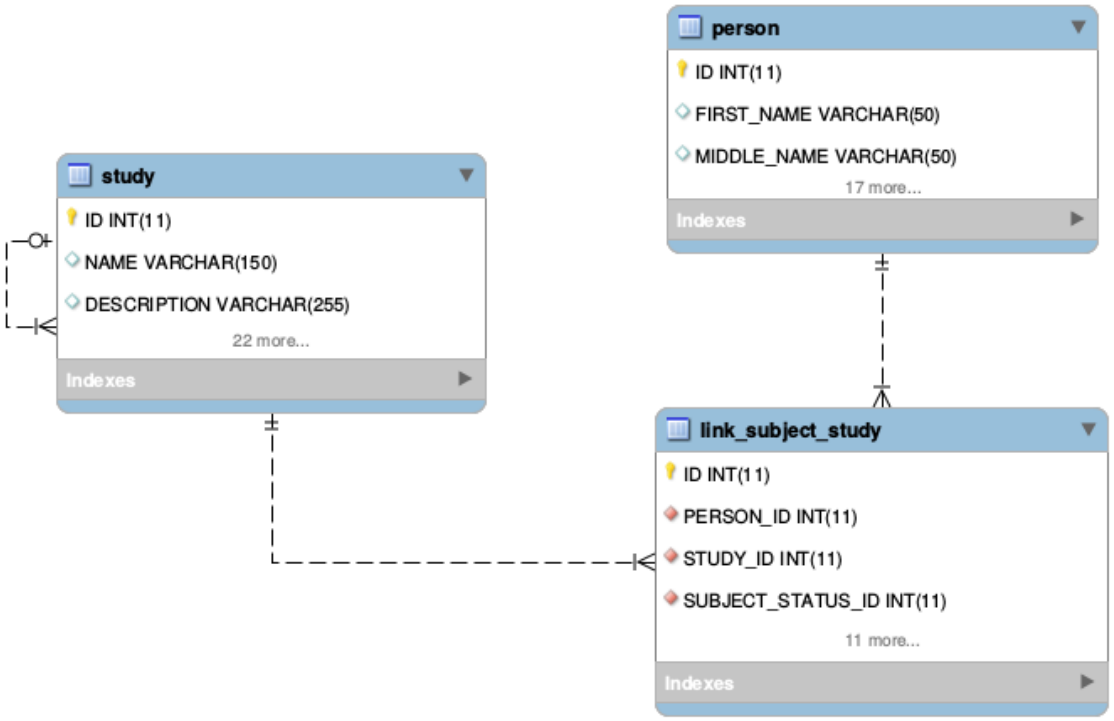


Figure 4.7: The Ark's primary table structure to map a person to a study. A person can map to a Study and its sub-studies via the foreign key relationships in the `LINK_SUBJECT_STUDY` table.

defined and an independent table structure to hold family relationships has been created. Relationships can be either parental or twin relationships.

The study table represents the study entity created inside the Ark and it has a self-reference to the sub-studies. The PERSON table represents the subjects who reside in a study. The `LINK_SUBJECT_STUDY` table uses a one-to-many relationship to represent a single person in multiple studies by the subject's UID. The table design in Figure 4.8 below enables single subjects to be shared in multiple sub-study entities related to a study entity. This is the backbone design of the Ark's table schema and hundreds of tables have been added with respect to this design.

Parental relationships are the backbone to the discovery of pedigree information for a subject in context. Therefore, the `LINK_PEDIGREE_SUBJECT` table records the parental relationships between study subjects who reside in the `LINK_SUBJECT_STUDY`

table. LINK_SUBJECT_ID represents the subject in context and has a foreign key reference to the LINK_SUBJECT_STUDY table. The RELATIVE_ID column has a foreign key reference to the LINK_SUBJECT_STUDY table and represents the relationship with the parental subject. Parent type is defined by the RELATIVE_ID foreign key reference to the RELATIVES table. The parent type is always either a father or a mother. According to this table structure, a family with two parents and three children would add six rows to the LINK_PEDIGREE_SUBJECT table. For each child the system records two parental relationships: one record for the father relationship and another record for the mother relationship.

The next important relationship type in a pedigree dataset is the twin relationship. Twin relationships are stored in the LINK_SUBJECT_TWIN table and are represented by the FIRST_SUBJECT and SECOND_SUBJECT columns, which link as the foreign key reference to the LINK_SUBJECT_STUDY table. The TWIN_TYPE_ID column refers to the TWIN_TYPE table to declare the twin relationship as monozygotic or dizygotic. This table representation avoids declaring a twin relationship per subject in context and maintains the twin relationship as a single entity to avoid data duplication.

A certain set of information is essential for pedigree visualisation and validation. The STUDY_PEDIGREE_CONFIG table contains the study-specific pedigree configuration options for age, date of birth and affected status in the AGE_ALLOWED, DOB_ALLOWED and STATUS_ALLOWED columns, respectively. Depending on the Boolean values presented in these columns, pedigree diagrams display or hide those attributes. The CUSTOM_FIELD_ID column is a custom field that holds the affected status of a pedigree subject, endorsed as either Yes or No. The FAMILY_ID column refers to a custom field that stores the Family ID value for the pedigree subject. Based on the Family ID value, store pedigree meta data records for subject in context. The INBREED_ALLOWED column allows the researcher to declare whether inbreeding relationships are allowed. If allowed, the system omits pedigree validations for inbreeding relationships when assigning parental relationships. Otherwise,

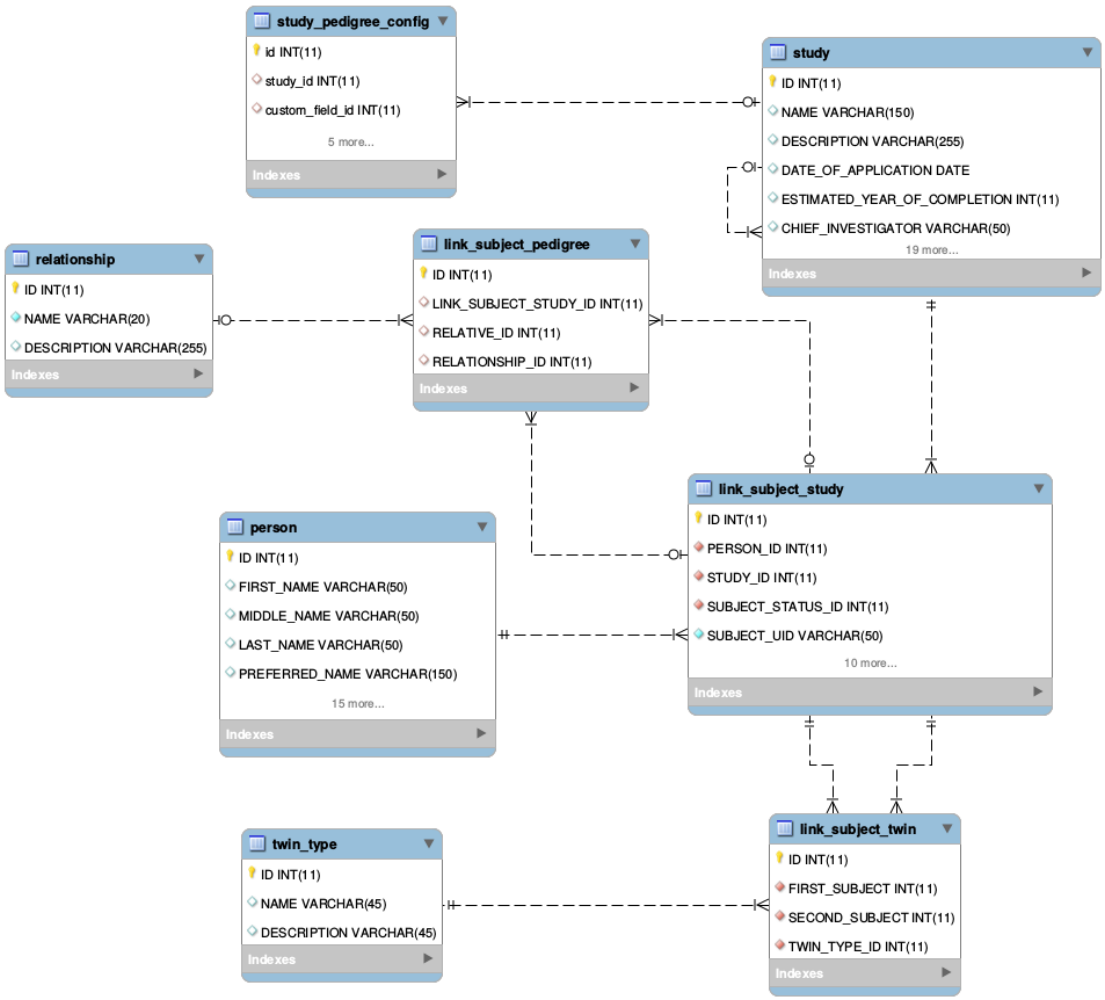


Figure 4.8: The Ark's pedigree module entity-relationship diagram.

the system would validate the new relationship against existing pedigree relationships for the presence of inbreeding.

The following section outlines an approach that avoids the limitation of recording relationships for each subject against all other family members. This approach gives the opportunity to record n-number of parental and twin relationships that are determined by the database limitations and discovered on an on-demand basis per subject in context.

4.6 Pedigree discovery

Storing pedigree datasets is crucial for data management of family studies. Regarding the study based pedigree data management it is equally important to discover the relationships for the subject in context, which the Ark can execute dynamically. When the subject in context is changed, the system is capable of re-populating the pedigree structures for a new proband.

I developed the BloodLine algorithm to discover the birth-related pedigree members for a subject in context. The first phase of the algorithm starts from the subject in context and traverses through the parental relationships, storing them for future reference. In the second phase, the algorithm searches children who descended from the parents identified in the first phase. After completing phases 1 and 2, the algorithm marks the twins with their twin type from the relative list.

The BloodLine algorithm is compliant with the best practices of an algorithm, including finiteness, definiteness, input, output and effectiveness [182]. Considering the finiteness of BloodLine algorithm, it first traverses the parent relatives and then searches for their children. Later it builds the twin relationships among the relatives. These steps are followed for any pedigree dataset introduced to the BloodLine algorithm and the execution order is consistent for all datasets. There are a finite set of steps declared for the BloodLine algorithm. For any given subject in context, a parent selection check is performed for both parents and child selection check is performed for children, irrespective of whether they have been selected as relative. If they have already been selected as a relative, the algorithm avoids adding them to the existing relative list to eliminate duplicates. The algorithm always accepts a UID of an existing Ark subject as input. This subject UID is used as the proband subject and the searches for parental and child subjects are based on the proband's UID. The BloodLine algorithm outputs the relatives list including the subject in context. The relatives list is not empty at any time because subject in context is always included. The effectiveness of the BloodLine algorithm has been tested

against real and hypothetical datasets.

4.6.1 Testing the effectiveness of the BloodLine algorithm

Effectiveness gives a credible assurance to the users about the accuracy of an algorithm. Hence, BloodLine was tested against many scenarios. For a real dataset, The Twins and Sisters study [183] was been used to evaluate the BloodLine algorithm's design. The Twins and Sisters study includes 5,120 subjects (with 544 monozygotic and 339 dizygotic twin pairs) that belong to 1,564 families. The BloodLine algorithm successfully processed the Twins and Sisters study's parental and twin relationships and was able to identify the family members for the study's subjects. In addition, the BloodLine algorithm was systematically tested for different family structures including large families (number of family members per family > 25), different multiple birth statuses (triplets, quadruplets, quintuplets, etc.) and families with consanguineous relationships. These tests were carried out by the developers and researchers who were interested in using the Ark pedigree module for their data management. Based on the above mentioned test outcomes, there is strong evidence to support the effectiveness of the BloodLine algorithm.

4.6.2 Complexity of the BloodLine algorithm

The complexity of an algorithm is based on the space and time taken to execute it. Algorithm space complexity describes the memory used during the execution. The BloodLine algorithm's space acquisition is based on the number of relatives discovered per subject in context. Referring to the algorithm pseudo code (see Section 4.6.3), the subject relatives list is populated according to the number of blood relatives identified. Therefore, there is a linear increase in memory usage per relative added to the list and space complexity rounds up for O(n).

Time complexity is based on the number of steps required to execute an algorithm. The BloodLine algorithm's execution steps are parallel with the number of

relatives identified per subject. According to the pseudo code (see Section 4.6.3), additional relatives create an extra iteration in the parental search and the child search. Therefore, there is a linear increase in time based on number of relatives and time complexity is declared as O(n) for the BloodLine algorithm.

4.6.3 BloodLine algorithm pseudo code

Algorithm 1: BloodLine

Input: A *subjectUID* of subject in context, A *studyId* of study in context

Output: List of *subjectRelatives* linked with the subject in context's blood line

```
1  relativesQueue ← ∅;
2  subjectRelatives ← ∅;
3  relativesQueue ← subjectInContext;
   // Search for parent relatives
4  while relativesQueue ≠ ∅ do
5      selectedRelative ←relativesQueue.getFirstElement();
6      parentsList ← searchParentRelativess(selectedRelative);
7      for parent in parentsList do
8          if parent is not identified before then
9              relativesQueue ← parent;
10     subjectRelatives ← selectedRelative;
11 relativesQueue ← subjectRelatives;
   // Search for child relatives
12 while relativesQueue ≠ ∅ do
13     selectedRelative ←relativesQueue.getFirstElement();
14     childrenList ← searchChildRelatives(selectedRelative);
15     for child in childrenList do
16         if child exists in subjectRelatives then
17             x ←subjectRelatives.get(child);
18         else
19             x ← child;
20             relativesQueue ← child;
21             subjectRelatives ← child;
22         if selectedRelative is a male then
23             selectedRelative is assigned as father to x;
24         else
25             selectedRelative is assigned as mother to x;
26 return subjectRelatives;
```

4.7 Pedigree validation

Pedigree validation requires the system to satisfy two criteria. The first criterion of pedigree validation is checking parental relationships against the gender, date of birth and optional consanguineous relationship restrictions. This eliminates the selection of invalid genders for parental relationships. The validation restricts selecting a female or an unknown gender type subject to be assigned as a father, and vice versa for the mother. Date of birth is also validated when creating parental relationship. When both the parent's and the child's date of birth are defined, the system validate those dates before assigning a subject for the parental relationship. This condition eliminates the creation of parental relationships with invalid dates of birth.

The second criterion of a pedigree validation system restricts consanguineous relationships. Consanguineous relationships are not rare when building large pedigree structures [184] but allowing consanguineous relationships could cause major data issues when creating parental relationships. The Ark pedigree module has its own consanguineous relationship validation mechanism.

The consanguineous validation is carried out based on the principle of graph data structure circular validation.

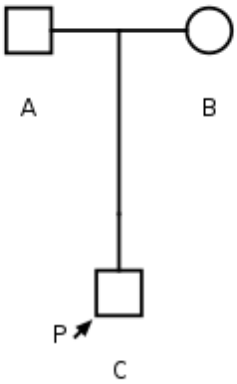


Figure 4.9: Nuclear family pedigree diagram.

The undirected graph representation of the nuclear family in Figure 4.9 is as fol-

lows. The pedigree members are represented by the vertices, and the relationships are represented by the edges. Therefore, the graph representation of the pedigree consists of three vertices added to represent the pedigree members and two edges to represent the father and mother relationships (see Figure 4.10).

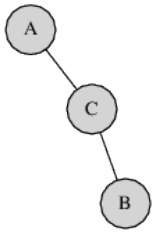


Figure 4.10: Nuclear family represented by a graph structure.

When a pedigree is represented as an undirected graph, it is possible to detect consanguineous relationships using graph-based circular validations, but there is an issue when representing non-nuclear family in a graph data structure. This is shown in the pedigree diagram in Figure 4.11.

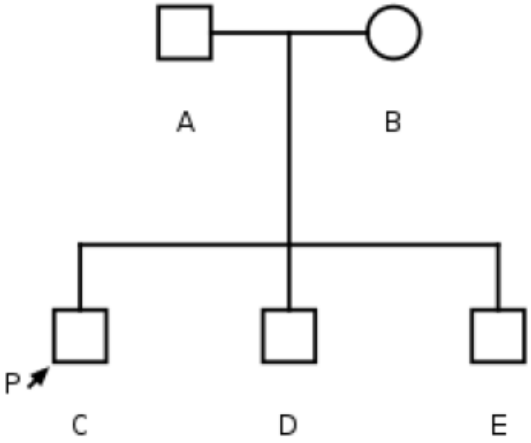


Figure 4.11: Sample pedigree diagram of two parents and their three children.

This pedigree consists of five individuals who are two parents and three sons. There are no consanguineous relationships among them. This pedigree can be represented using the graph data structure shown in Figure 4.12. There are five vertices

to represent the family members and six edges to represent their relationships. This is an incorrect representation of the family because there are three circular relationships: ADBE and AEBC. To overcome this situation, an extra node for each parent relationship pair has been introduced. The new graph representation is shown in Figure 4.13.

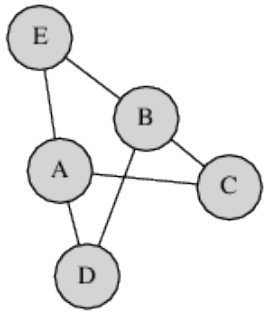


Figure 4.12: Two parents and three children represented by a graph structure. The graph consists of five vertices and six edges. Closely observing the diagram can identify two circular relationships.

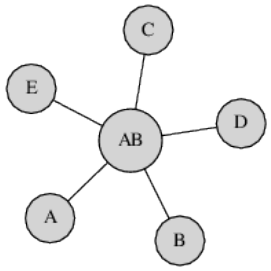


Figure 4.13: Enhanced graph presentation of a family with two parents and three children with an extra vertex to represent the parent relationships.

The new graph representation consists of six vertices for the family members and five edges for the relationships. AB is introduced as an extra vertex to act as a bridging node to represent A and B's parental relationships to the children C, D and E. This additional parental vertex prevents the introduction of circular relationships for non-consanguineous family pedigrees when using graph structures to represent the pedigree. The examples above show families without consanguineous

relationships. A family with a consanguineous relationship is shown in Figure 4.14.

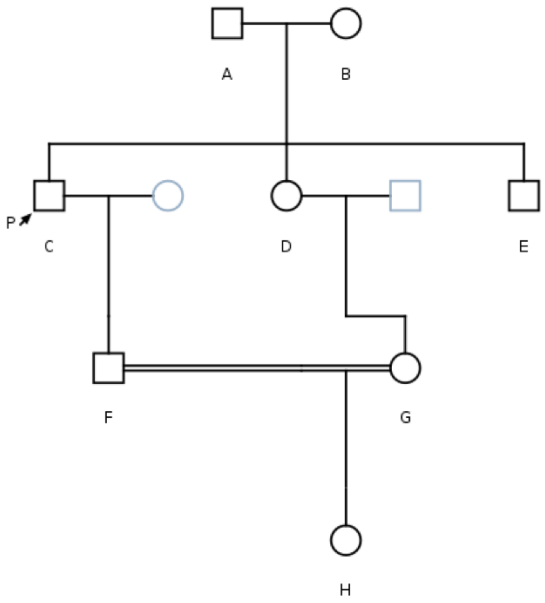


Figure 4.14: Pedigree diagram of a consanguineous family.

According to the pedigree representation in Figure 4.14, the consanguineous relationship has been created by subject H. Representation of the above family in the graph data structure in Figure 4.15 shows a circular relationship between the vertices. When the BloodLine algorithm is executed for the subject in context, graph circular validation is applied to identify inbreeding.

4.8 Pedigree visualisation

Pedigree visualisation is an important part of pedigree data management, but the task is not a straightforward due to the complex relationships between individuals. A number of pedigree visualisation tools have been developed and the pedigree task force has standardised pedigree data representation through a set of standard symbols to visualise data elements. A H-tree is a popular visualisation layout that is used to display relationships in a pedigree diagram [172] as a fractal tree struc-

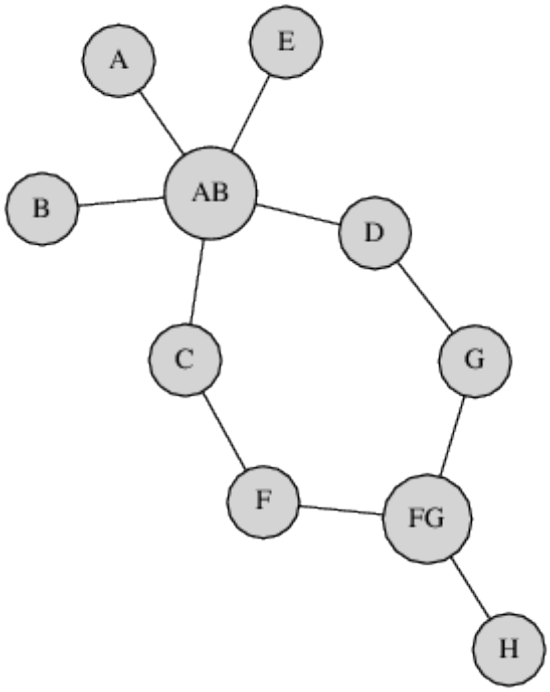


Figure 4.15: The Ark representation of consanguineous family with extra vertexes. One circular relationship to represent consanguineous relationship can be observed.

ture. The H-tree uses optimal space management and is suitable for exponentially growing pedigree structures.

I selected the Madeline pedigree rendering engine to visualise the relatives identified for a subject in context [181]. The subject in context's relatives are identified through their parent relatives by the BloodLine algorithm. The visualisation engine is interoperable with the subject's relative dataset and is able to display complex family relationships.

The Madeline pedigree visualisation engine is capable of visualising complex family structures. This includes large families with multiple generations of family relatives or consanguineous relationships. Madeline's pedigree drawing algorithm avoids line crossing in complex pedigree structures and uses graph-based hybrid algorithms to draw the pedigree. The Madeline software is released under the GNU General Public License and has attracted substantial attention from the open-source

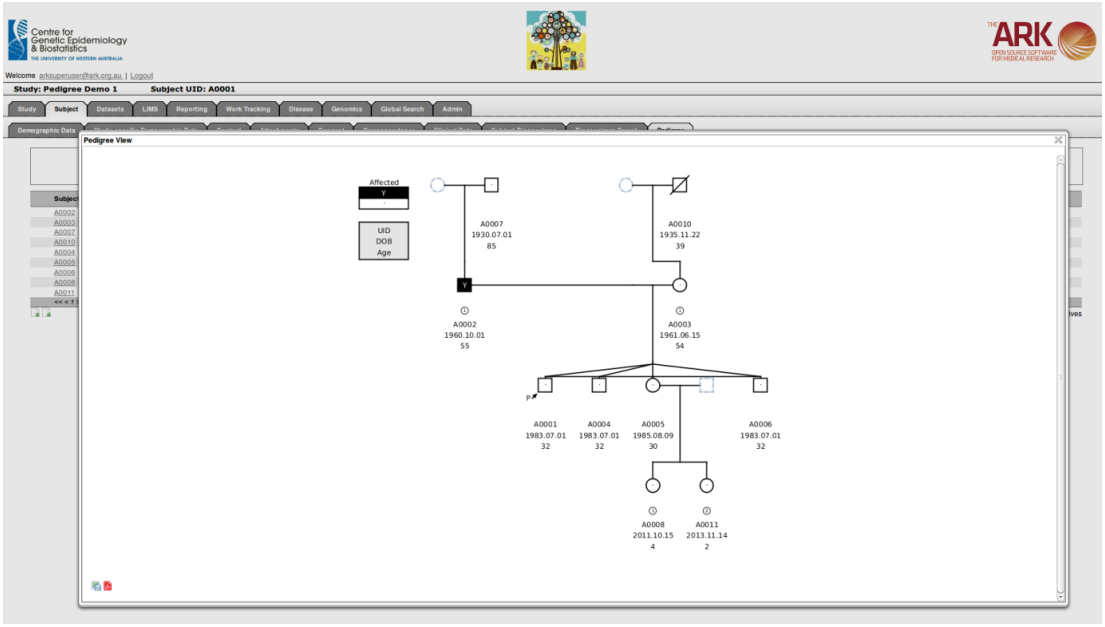


Figure 4.16: The Ark pedigree display pop-up dialog with a pedigree structure and export buttons to download the pedigree diagram in PDF or PNG formats.

software development community. The source code is available for download and is free to use in other open-source software projects. The software implementation is based on C++.

As input, Madeline takes pedigree data formatted according to a given template file. The rendering engine is capable of validating the input data against a set of rules before processing its content. The validated file is then processed and converted to a pedigree representation. The output is generated in the extensible mark-up language based scalable vector graphics format and can be rendered in image viewers or browsers that support scalable vector graphics. The extensible mark-up language output gives end users freedom in viewing the results as well as editing the content. Representational state transfer web services support the extensible mark-up language output and give developers the flexibility to use the output over the hypertext transfer protocol.

Figure 4.17 demonstrates the standard input file for the Madeline software.

```
FamilyID
IndividualID
Gender
Father
Mother
Deceased
Proband
DOB
MZTwin
DZTwin
Sampled
Affected

FA0002  C  M  A  B  . Y 1963.07.01 . . . .
FA0002  D  F  A  B  . . 1964.08.05 . . . .
FA0002  E  M  A  B  . . 1965.09.12 . . . .
FA0002  A  M  .  .  . . 1930.10.01 . . . .
FA0002  B  F  .  .  . . 1931.06.15 . . . .
FA0002  F  M  C  .  . . 1981.04.15 . . . .
FA0002  G  F  .  D  . . 1982.03.15 . . . .
FA0002  H  F  F  G  . . 2005.06.15 . . . .
```

Figure 4.17: Sample Madeline input file format

When the pedigree data input file is submitted to the Madeline software, it will output a scalable vector graphics pedigree file. The Ark was developed using the Java language and all of its base frameworks use the Java Virtual Machine (JVM). The original design of the Ark is to interact with third parties by using web services or by including their Java Archive files as dependencies to the system. In the case

of Madeline, non-Java based tools needed to be integrated to synchronise with the Java runtime threads.

Java Native Interfaces (JNI) is the preferred methodology to interface with non-JVM based applications. The JVM directly calls native methods using the existing JNI method declarations. JNI methods then communicate with native methods. The actual process execution is independent of the JVM and only provides input parameters for the execution. Similarly, the output is interpreted to JVM and results are input to the JVM memory heap. The advantages of this approach are that it provides a mechanism for communication with non-Java based libraries, preserves legacy code and saves the development time needed to recreate existing well-tested functionality in non-Java programs. The disadvantages of this approach include hard handling exceptions, which could break the existing JVM, and security vulnerabilities in native libraries.

The default communication media for the Madeline engine is file input and output streams. Integrating Madeline to the Ark required building a secure communication mechanism. Also, file streaming is not a thread safe mechanism and adds dependency to the deployment environment. Byte input and output streams were chosen to communicate with Madeline. To support the byte input streams, existing Madeline source code was modified and the Madeline library was re-packaged. Also, a set of wrapper classes were introduced to the Ark service packages to accept the byte input streams.

The pedigree diagrams can be downloaded in either PDF or PNG or SVG formats. In addition to the visual representation of pedigree diagrams, researchers can download data files that are formatted according to Madeline and Ark format specifications. The data files are useful for importing the pedigree datasets to external resources, including third party pedigree utilities, different pedigree rendering engines and quality control checking.

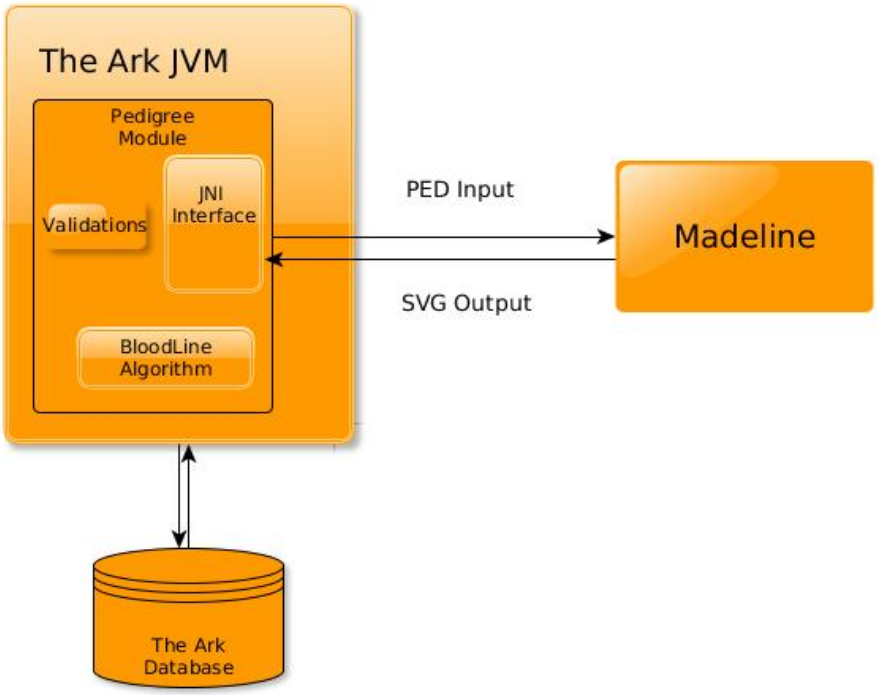


Figure 4.18: Architecture diagram of the Ark pedigree module integration with Madeline library

4.9 Future work

Different cultures express pedigree relationships differently and there is no universal language for relationship names. Therefore, an external rule engine will be introduced as part of the future work to declare relationship names. Introducing an external web service is useful for third-party vendors who need to communicate with the Ark system and access the pedigree information (visualisation and data).

4.10 Conclusion

This ' has presented the Ark's pedigree data module and its capabilities. A data model has been introduced to manage pedigree datasets in a study-subject based data environment. Based on the data model, the BloodLine algorithm is developed

to dynamically discover the birth-oriented relatives per subject in context. Representation of pedigree data in a graph data structure enabled detection of consanguineous relationships using circular validations. A JNI pipeline to Madeline enables visualising pedigree structures discovered from the BloodLine algorithm for the subject in context. Bulk uploading enables importing large family datasets to the Ark with validation of consanguineous relationships.

The Ark pedigree module provides a complete data management solution to handle pedigree datasets. The next ' focuses on the software architectural approach to handle high-performance computing.

Enabling the High-performance Computing for web application by microservice architecture.

5

A Super Computing Pipeline for the Ark (SPARK)

5.1 Introduction

This chapter introduces a novel software architecture design to enable high - performance computing (HPC) and big data management for the Ark biomedical data management system. As became evident in previous chapters (the Literature review and the Ark chapters), there is a significant need for enhanced capability to manage genomic datasets inside existing study-based phenotypic biomedical data management systems. To address this challenge, this chapter discusses a novel software design approach to manage genome-wide association study (GWAS) datasets

through implementing an analytical platform supported by massively parallel HPC.

Section 5.2 examines the motivation and design goals of the software design approach to enhance the Ark's biomedical data management system capabilities to suit GWAS datasets. The popularity of GWAS and the importance of their findings has encouraged global research collaborations and necessitated advanced software systems to manage the data. The following high-level design requirements were identified: (1) provide the health and medical research community with user-friendly access to the massively parallel HPC resources, (2) bring state-of-the-art complex genomic data management to the international GWAS community and (3) foster collaboration among the international GWAS community. These design goals informed the software design and implementation process.

Section 5.3 discusses the spectrum of software design architecture that is capable of meeting the software design goals. These software design goals include: (1) distributed service identifier management, (2) efficient genomic data management with distributed services, (3) storing and sharing computational packages for use in analysis and (4) executing analysis and sharing the results. Based on these design goals, the evolution of the software architecture designs is discussed. After systematically benchmarking the existing software architecture guidelines (monolithic, N-tier and service-oriented), microservice architecture has been chosen for the Super Computing Pipeline for the Ark (SPARK) system.

Microservice software design is discussed in the SPARK architecture section (Section 5.4). This includes the inherent advantages of the microservice architecture selection for SPARK. The SPARK system consists of the micro service component to manage the distributed service end points, a purpose designed genomic data management component for GWAS data, a computational package component to store existing analysis packages, and analysis and results in a component to conduct the experiments based on selected data source and computation package. The analysis component is capable of storing and sharing the experimental results. The Ark's Genomic Module has provided the web interface for the following components

design in SPARK module via the web service interface.

Finally, Section 5.5 presents a technical evaluation of the SPARK microservice architecture and individual component design. Software experts' recommended software quality attributes (security, availability, interoperability, modifiability, performance, testability and usability) are evaluated and discussed against the SPARK design.

5.2 Motivation and design goals of SPARK

High-performance computing is an expensive commodity for processing and analysing complex data. Despite the expense, researchers are requesting more computational power for processing large datasets. Large volumes of data are generated from modern research, and these datasets include complex data elements. In genomics research, high-performance computational power is essential, especially for analysing GWAS datasets that include millions of Single Nucleotide Polymorphism (SNP) pairs from thousands of subjects.

From a software architecture perspective, several attempts have been made to use high-performance computational power for genomic data management and analysis. While software systems have been built to support the GWAS data management requirements of individual studies, collaborative GWAS by large consortia require a common research platform. Facilitating collaborative research based on GWAS poses a challenge to software developers.

Based on the above requirements, a set of design goals was formulated for the SPARK project to facilitate common GWAS data management and analysis via the Ark.

5.2.1 Provide health and medical research community with user-friendly access to massively parallel computing resources

Implementing a system to manage complex GWAS datasets requires a user-friendly data management system for researchers. Accessing HPC resources requires computer literacy and hands-on experience in working on a similar project. Gaining the required level of HPC literacy is extra work for GWAS researchers and adds an overhead to already complex GWAS analysis. Genomic data management is quite different from the management of ordinary datasets as they usually contain gigabytes of data and complex data relationships.

Genomic data management requires a special set of data management techniques and a specialised set of skills. Ordinary relational databases are not suitable because genomic datasets exceed the capacity of relational database management systems. GWAS researchers depend on data managers who possess the specialised skill set to manage GWAS data repositories. Hiring additional data managers increases the overall cost of GWAS project management.

A user-friendly HPC dashboard and simplified genomic data management system is a crucial requirement for modern GWAS. Well-established software architecture design has been required to support the HPC power for a software system. In addition, carefully benchmarked database environment is required to embed the genomic data management for an existing phenotypic management system.

5.2.2 Bring state-of-art complex genomic data management to the international GWAS community

GWAS datasets are considered big data because of their size and complexity of information. Ordinary data management systems are unable to deliver the expected outcomes regarding space efficiency and query optimisation for large genomic datasets.

When relational databases were first introduced to computer systems, the concept and size of genomic datasets were not yet known. Relational databases have limits to the number of rows per table. Genomic datasets easily exceeded the capacity of a single relational database. As an alternative, non-relational databases have been introduced to overcome the limitations of the relational data environment. The SPARK data architecture will be discussed in the next chapter with further technical information.

SPARK aims to introduce a sustainable data environment to reduce the burden of managing genomics datasets. The SPARK approach consists of software architecture to provide computational power for data processing, sharing computational packages, analysis of existing datasets and management of datasets. SPARK also provides a user-friendly management environment for GWAS researchers.

5.2.3 Foster collaboration among global GWAS community

Collaboration – through sharing data and analysis techniques – is essential when designing a successful GWAS. To facilitate collaboration, an open-ended platform that is accessible by genomic researchers worldwide is required. Some existing approaches satisfy the requirements for collaboration and include those developed by commercial vendors and research groups. The Galaxy project, PLINK GWAS tools and SeqWare (see Sections 2.2.3 and 2.4.2) are examples of collaborative genomic data management and analysis platforms.

These systems provide data harmonisation methods, analysis techniques and custom application components. Security is a key issue that has been addressed by these systems to enable protected workspace for global researchers to carry out their work. The systems provide application interfaces to program customised plug-ins and guidelines to work with multiple data formats. Given the existing workflows, it is essential to provide a flexible system with an interface with different

hardware platforms. Enabling big data management techniques and web-based access are fundamental components of a collaborative GWAS. It is essential to establish software architecture that caters to the collaborative GWAS.

The three design goals described above formed the project vision: "To create a leading open source knowledge discovery platform that leverages massively parallel computational power for heterogeneous GWAS data sets". This required choosing a supporting software architectural design.

5.3 Candidate architectures

Establishing a solid architecture for the SPARK system is a key research goal of this PhD work. Before implementing an architectural guideline for the SPARK system, it is important to understand the key functional aspects that are necessary to satisfy the project goals (see Section 5.2).

Distributed service identifier management

The SPARK project has to be compliant as a distributed service location mediator facility. Therefore, the system needs to record service locations and their access credentials. When this information is recorded, the system navigates to the service and executes the required task. The idea behind this concept is to initiate the SPARK nodes with collaborators and manage the functionality based on a predefined set of independent services.

Efficient genomic data management with distributed services

The datasets to be facilitated by this system are mainly GWAS datasets. Introducing an efficient genomic data management approach will be discussed in Chapter 6. The novel data management approach requires supportive software architecture. SPARK consists of lightly coupled service-oriented workflows. These

include traversing to distant data sources and analysing their content. Another objective is to identify the pre-emptive file structures (e.g. PLINK PED/MAP and binary file formats) and process the data based on the file format. Querying is an important aspect of this type of data centre and enabling query result extraction is a key objective of the data management architecture.

Storing and sharing computational packages

Sharing computational analysis packages is one of the major objectives of SPARK. The principle of this objective is to develop a mechanism whereby computational packages can be shared and executed on multiple datasets. To satisfy this requirement, the SPARK system needs to establish a centralised repository of computational packages that contain a specific GWAS analysis technique. An analysis technique can be a single program or an execution pipeline specific to a computational facility. If the computational package is program source code, it needs to include instructions for compiling and execution. If the computational package only contains an executable program or an analysis pipeline, then execution instructions are required.

Computational packages must include instructions on how to use the package. As with software application servers, the SPARK system required a system-specific meta information file to provide handling instructions. The computational package must be in an archived format to reduce the physical storage space. This archiving mechanism has to be compliant with the widely accepted ZIP format, which has been selected as the SPARK compression type.

Executing computational analysis

SPARK provides a simplified dashboard to manage data sources and computational packages. Analysis can be run at a selected service point described in distributed service identifier management with compatibility to execute in the environment

bound to the computational package. The meta information excludes the data source selection and needs to be specified for each analysis via the web interface. The analysis runs as a background job and its status is updated on completion. When an analysis is completed, researchers can download the results according to the result output format defined in computational package. Ultimately, researchers would be able to run a single computational package for multiple datasets and compare the results.

For the requirements and design goals described above to be satisfied, there needs to be solid software system architecture design to enable SPARK functionality. Therefore, the previous software architectural designs as well as the modern best practices were systematically analysed to choose a best-fit design for the SPARK goals. The following software architecture designs have been considered and evaluated against the SPARK design objectives.

It is important to understand the context of software architecture before discussing architectural approaches. According to the book Software Architecture in Practice [185], software architecture is a set of structures that describe a system through a set of elements, the relationships between the elements, and the properties of the elements and relationships. A software structure is developed as a blueprint or abstract for the system implementation, which helps to maintain unique design in the entire system.

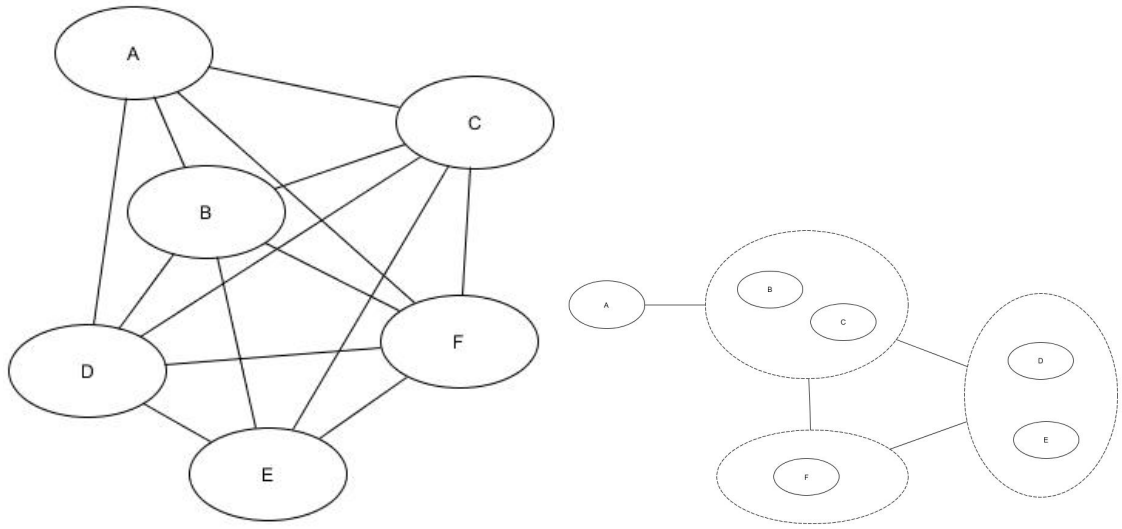
When implementing a software system, it is necessary to use a solid architecture to ensure reliable system implementation. Therefore, the SPARK system design phase must examine the approaches of previous architectural models and the availability of design techniques. Examining the history of software systems has indicated at different architectural approaches has been followed over time by the designers [186]. The main drivers of changes to software design are the evolution of technologies, fast growing user expectations and competency of the development communities to produce better software design.

5.3.1 Monolithic software architecture

In the early days of software systems, there were no separate database systems for data management. Software applications had their own data management mechanisms that were most probably similar to modern database management systems [187]. These systems were tightly coupled to the data elements and hard to change only in a single place in the source code.

A lack of understanding of best practices for software architecture, little demand for improvements and few developers involved in developing and managing the code base has prolonged the use of monolithic systems. As computer technology improved and demand for software systems increased, the need for novel architectural approaches to deal with known drawbacks in monolithic applications also increased. The requirement for re-usable components has driven software design towards flexibility and extending design techniques towards much more agile design [188]. Even though some software systems with monolithic architecture have survived, others have fallen into a design description of a big ball of mud-type architecture (See Figure 5.1a) or more prudent application development with a single application instance. Figure 5.1 shows tightly coupled software components (A, B, C, D, E and F) in a monolithic system (Figure 5.1a) and in a better organised single application design (Figure 5.1b). Both of these examples have shown high coupling between the application components and operating as a single application instance.

PLINK tool set (see Section 2.4.2) is one of the most widely used applications for GWAS research and its implementation has a similar approach to the monolithic design. As a standalone tool, the PLINK implementation comprises data management, analysis techniques and result extraction in a single codebase. When examining the PLINK implementation it is clear that its design has not followed a specific layered approach to group the functionality to different modules or services. Integrating the PLINK functionality with another software system or changing the existing functionality requires implementation on all of the integrated



(a) **(b)**

Figure 5.1: Monolithic architectural approach of the application design.

solutions rather than a single implementation. PLINK is a standalone tool, and the monolithic architecture has operated as command line-based application. Further enhancements, such as integrating with web services and developing user interfaces, will be complicated.

5.3.2 N-tier architecture

The popularity of relational database management systems has created N-tiered application architecture. Unlike the monolithic systems, N-tiered architecture uses separate application layers for specific tasks [189]. The early application of N-tier software architecture involved an application structure in three layers. The principal data layer allows communication with the database and handles the transactions. The business layer manages the functional logics, and the presentation layer handles the user interface.

N-tiered architecture has enabled developers to maintain a clear code base that is easily managed and has helped to develop set of common software components

that can be used throughout the application. There is a significant consideration raised by the software development community regarding the data communication between the layers of the application design [190]. This allows developers to maintain an abstract design for each layer and streamline communication between application layers.

The Ark (see Chapter 3) is a typical example of N-tier software design architecture. The application functionality is organised into the web tier, service tier and data tier. The implementation of each tier reflects the specialised set of responsibilities assigned to it. The Ark modules follow the N-tier design and implementation is carried out with respect to the N-tier design principles discussed earlier in the section. The web tier represents the user interface design (web page layouts) and presentation logic for the user input and output. The service tier represents the logic behind the processes which are implemented to support the functionality expected by the system. A typical example is the pedigree discovery algorithm (BloodLine) implementation in the study module service tier (see Section 4.6). The data tier represents the mapping between the database tables and the application implementation. In addition to table mapping, the data tier represents the database transaction handling to prevent data loss in the database operations (create, retrieve, update and delete). The N-tier based design has increased the code clarity of the Ark implementation and increased the performance by following the spring framework-based dependency injections. Figure 5.2 illustrates the high-level view of the Ark's design, which uses N-tier architecture by organising the implementation into the presentation, service and data tiers with bi-directional communication pipelines.

5.3.3 Service-oriented architecture

Service-oriented architecture was proposed along with the introduction of the internet. Compared with the previously described N-tier software architecture, service-

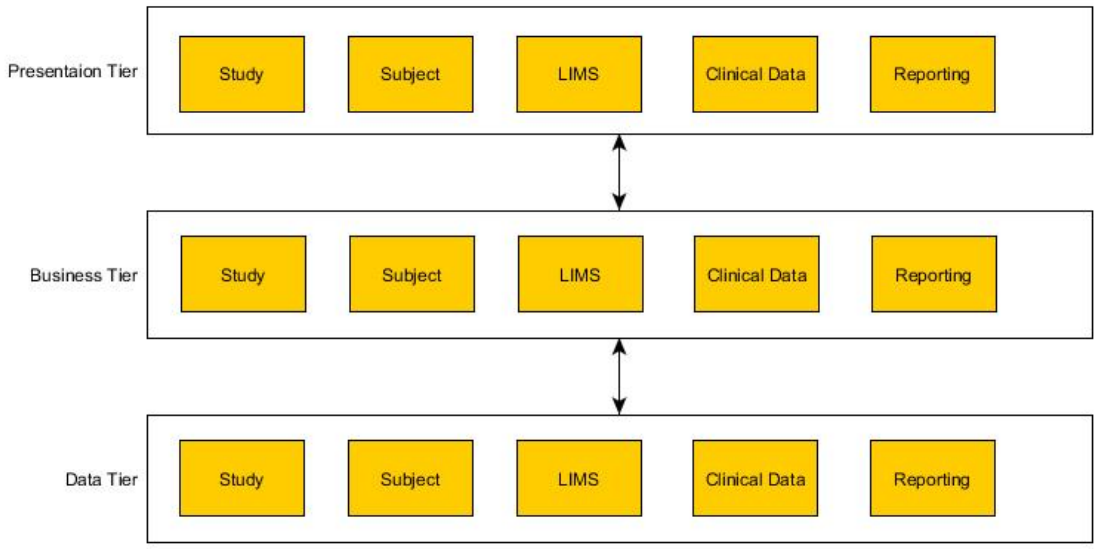


Figure 5.2: The N-tier architecture of the Ark system.

oriented architecture has provided an independent and agile development framework for software developers [191]. N-tiered software systems are tied to a single location and application runtime is limited to single execution container. The introduction of the internet has demanded more scalable application development.

To increase application stability based on the web protocols, top priority goes to the hypertext transfer protocol-based standards [192]. The hypertext transfer protocol is a widely supported web protocol that is supported by network controllers (routers and switches) across the globe. Based on the internet standards, different business functionalities are developed into individual web services in application developments. Individual business functionalities are amalgamated into a single internet-based service described as Web Service Description Language (WSDL).

Here, individual web services are considered as the components of a software application. These services operate independently and do not depend on each other to complete a transaction. The web service components implemented are specific to their standards and called up by the application according to the requirement. The N-tier based applications consist of presentation components to handle the user

interactions and communicate with other services according to the user specified interaction scenario.

The N-tier design enables the system to operate in a fail-safe environment. Components can be deployed in different geographical locations and managed independently in different application execution environments. This gives developers the freedom to manage system components independently based on module environments [193].

SeqWare is a typical example of the service-oriented architecture (see Section 2.2.3). The design of the SeqWare system consists of a service module and user interaction modules. The user interaction modules consist of the command line module and a web module to interact with the service module. The service module communicates with the HPC sources to manage genomic datasets and execute analysis based on existing methods. The web module and the command line module both interact with the service module via the web services and trigger functionality according to the service calls. The N-tier based SeqWare design has given the opportunity to implement standalone execution of the service module and provide an extra security layer to hide the service implementation from the user interactions. Changes to the service module only affect the web services rather than the input modules. Therefore, the SeqWare service module has a common web service interface, and specific web calls mapped to individual functionality (see Figure 5.3).

Compared with the N-tier application architecture, service-oriented architecture has used the power of internet communication at the application level. It has also allowed developers to design applications independently and manage application functionalities in unique execution environment.

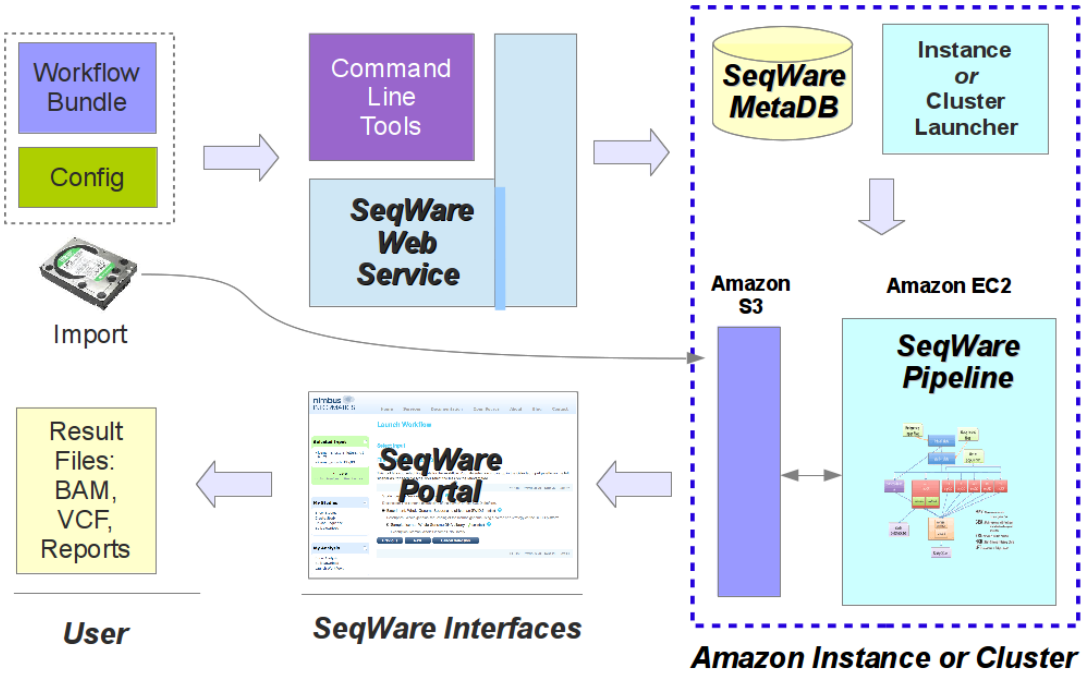


Figure 5.3: Service-oriented architecture-based design and system service calls diagram available in the SeqWare documentation (reprinted from [2])

5.3.4 Microservice architecture

The microservice architecture was designed to enhance service-oriented architecture capabilities [194]. Microservice architecture enables individual system functions to be operated as independent workflows. These independent workflows can have a unique design to represent the functional specification and are non-dependent on the other workflows. Two main design principles are followed when implementing microservice software architecture [49].

Decentralised data management

Compared with service-oriented architecture, micro services use decentralised data management [195] that is oblivious to the locations of the data. This is a novel way of managing the data compared with service-oriented architecture. Individual

services are created for different functional components; in the end, services tie up to multiple databases to manage the information.

In the micro service components, data decentralisation aspect has created a huge requirement to manage the datasets independently. This gives developers the freedom to create unique data schemas that do not use foreign key constraints to manage the datasets. This helps to implement fail-safe data storage in application data warehouses. Present day application functionality specific databases are introduced to the software developers to maximise the usability of data storage engines. Microservice architecture has maximised the efficiency of data storage.

Decentralised governance

Micro service components are not bound to a single governing policy [196]; instead they are independent of each service instance and can have a component-specific management style. Software governance leads to define best practices in various stages of the software development lifecycle (design, development, testing and deployment) or common guideline to every development iteration. The governing mechanism would be unique to a software development team and implemented to maximise the quality of software product [197].

A software governance mechanism includes selecting the software development methodology, establishing standards for design reviews, implementing code review standards, test review coverage and document review. It is helpful to have a unique set of software governance rules for a distributed software development environment. This is ideal for microservice software architecture because each service has its unique implementation mechanism, methodology, language, coding, testing and deployment environment.

A decentralised data management environment and a decentralised governing mechanism help to develop a unique software product. Microservice architecture has been described as evolutionary architecture [198], where the software is devel-

oped to meet specific requirements and is open to changes. Micro service software components are independent of each other and microservice architecture design helps to continue simultaneous development in various software components of a single application. Changes in one component do not affect other components because they are operating independently. This gives developers the freedom to easily replace problematic software components without affecting the usability of the total system. This type of software architecture is ideally suited to SPARK's design goals and system requirements.

5.4 The SPARK architecture

The SPARK architecture section aims to elaborate the implications of applying previously discussed software architectural design paradigms (see Sections 5.4.1, 5.4.2, 5.4.3, and 5.4.4) to the SPARK design and chooses a most suitable design. Applying the monolithic design to the SPARK project is extremely challenging due to the existing N-tier design of the Ark and its current development best practices. Adopting monolithic architecture design to SPARK could use either an existing MySQL database or another database to manage the datasets because the monolithic software approach needs to create a database connection to selected databases for existing application layers based on the design specification. Also, changes to SPARK must be implemented sequentially to the existing Ark implementation source code, by integrating a new set of classes and methods to enable monolithic design. The monolithic architecture facilitates designers to integrate changes to an existing application, saving the development time of building separate applications.

The disadvantages of using monolithic architecture for the SPARK system include light code cohesion, tight coupling between the components, restrictions to amending the existing software architecture, inability to maintain a fail-safe operation capability in the runtime, and reducing the operational capabilities of the system to interact with multiple HPC sources. Light code cohesion and tight cou-

pling mean that the individual software components depend heavily on each other without proper functional flows. In this case, a simple change to a single component requires modifications in multiple places, and the impact of the change must be tested in every dependent component.

Implementing a modest software system that is suitable for a HPC engine requires a specialised set of architectural best practices to cope in application processing. Software system with HPC hardware is supported by a set of specialised functionalities and incorporating them into the software system requires following modern architectural best practices including queue management, job submission, monitoring and fail-safe operations. This type of implementation is difficult in monolithic systems, and in particular, monolithic design-based applications are hard to implement as a fail-safe application because the application execution is based on a single container and unable to manage a system crash at runtime. Each HPC system uses unique programming specifications, and monolithic implementation requires addressing these specifications individually rather than generically. It is unrealistic that a system will be developed to meet future HPC requirements, and introduction of a new HPC requirement involves remodelling of the exiting application implementation.

The SPARK system will be developed as a set of new application layers to the existing Ark software. An application layer will be introduced to handle the HPC sources. A separate big data tier will be introduced for genomic data management, and data transaction layers for specific data management systems will be used to manage the heterogeneous genomic datasets.

Advantages of using N-tiered architecture for the SPARK development include high cohesion and less coupling between the application components. The N-tier design reduces the risk of breaking the existing application functionality by introducing separate operational layers for different application purposes, clear code structure for ease of developments and less code specification to developers to carry out the individual developments. Due to the layered architectural approach, system

implementation is broken down into layers based on generic application responsibilities. Therefore, N-tier design tends to minimise impact on tight coupling and less cohesion between the application layers. The SPARK system requires clear code structure to continue as an open-source project. N-tiered architecture has helped to maintain specific code segments to match up with the system goals. For example, the SPARK system will be implemented with separate layers for big data management and HPC. The layered software implementation helps developers focus on a specific software development point in the code branch. Rather than working on several parts of the software code base, developers can focus on specific application tiers related to the functionality.

The main disadvantage of the N-tiered software approach for the SPARK system is centralised application management. According to the design goals, the SPARK system is expected to be deployed in multiple locations with a common web interface to manage the resources. A centralised code base was needed to manage the changes in different aspects of the system. This has destabilised the independent system component management, which is a key requirement of the system. There has been a significant requirement to interact different HPC resources by the SPARK system. Therefore, using N-tiered application architecture for the SPARK system would require extra hardware to manage the computing power and analysis processes. The application performance depends on a single network bandwidth due to the single container approach in the N-tiered application.

The SPARK system and the service-oriented architecture would generate a clear light on its original objectives (see Section 5.2). Service-oriented architecture defines an independent set of components that operate independently and communicate with each other based via web-based protocols on a common set of objectives. Services are defined to represent the original objectives of the system and delegate service management by the original owners. Every component is mapped to a service and executed via the particular service end point. Therefore, SPARK was implemented based on the service-oriented architecture using the following guide-

lines.

According to the service-oriented architecture, SPARK would be developed with specific services to manage data sources, computation and analysis. These services would operate independently of the existing Ark implementation. The decentralised governing structure unique to each service enables the owners to manage their independent technical stack unique to the service. This will meet the initial objectives of the SPARK project and enable large collaboration among GWAS studies.

Although the service-oriented architecture approach satisfies the primary requirement of the SPARK project, it has one major problem, the replication of the service for the different entities conducting GWAS research. SPARK could not be limited to an existing service component serving multiple end points or HPC facilities. In such a case, it is required to replicate service end points to match study locations and would create N service definitions for each service. This would lead to considerable confusion among the system users when selecting the appropriate service to execute a task. Also, N number of services adds more coupling in the existing source code to manage multiple service ends and it is hard to select the appropriate service to manage an analysis.

With an understanding of the service-oriented architectural approach limitations, SPARK has been developed based on microservice software architecture. The original system implementation identified the initial service end points and declared initialisation points to locate the services. This has enabled multiple vendors to implement the SPARK nodes while following the Ark Genomics module Application Programming Interface (API) to be compliant. To support the SPARK implementation objectives, micro services are defined to initiate the URL compliant with each module. According to the principles of microservice architecture, SPARK modules were designed to operate independently from the Ark container and enable communication using the Representation State Transfer (REST) protocol. This has enabled the SPARK modules to operate independently with a unique set of operational aspects.

To implement a common web interface for the GWAS research community, the Ark Genomics module declares service end points in the micro service section. Declaring service end points marks their locations in the system and checks their availability. Therefore, accessing a data centre, installing a computational package and conducting an analysis will be carried out according to the independent implementation of the micro service owner. The independent implementation of data centres, computational packages and analysis are common in GWAS research infrastructure. Any research platform has its own set of system requirements and development best practices. Implementing a collaborative GWAS management system for global research studies needs to consider this factor very carefully.

The architectural diagram in Figure 5.4 explains this in a graphical representation. The diagram shows stated unique implementation of the SPARK nodes, which can be any number depending on the number of development instances. Each independent spark node can have its own implementation of data storage access methods (secure shell (SSH) and secure file transfer protocol (SFTP)) and a unique way of interacting with the HPC resources. The only condition is to follow a REST API to communicate with the Ark Genomics module, which is the only accepted communication protocol to integrate the SPARK node to the main system.

This type of implementation has allowed architectural best practices to be integrated with the SPARK design. Based on the design goals explained in Section 3.2, The Ark Genomics module was designed with four major components. The Ark Genomics module was developed and embedded into the Ark system following the microservice architecture. Independent SPARK nodes are integrated with the Ark Genomics module via the web services. The Genomic Module provides a common web interface for the encapsulated SPARK services with the support of the existing Ark user interface design. Therefore, the Ark Genomics module provides an interactive web interface for Ark users to work on encapsulated SPARK services without revealing the complexities and design localities of the infrastructure based

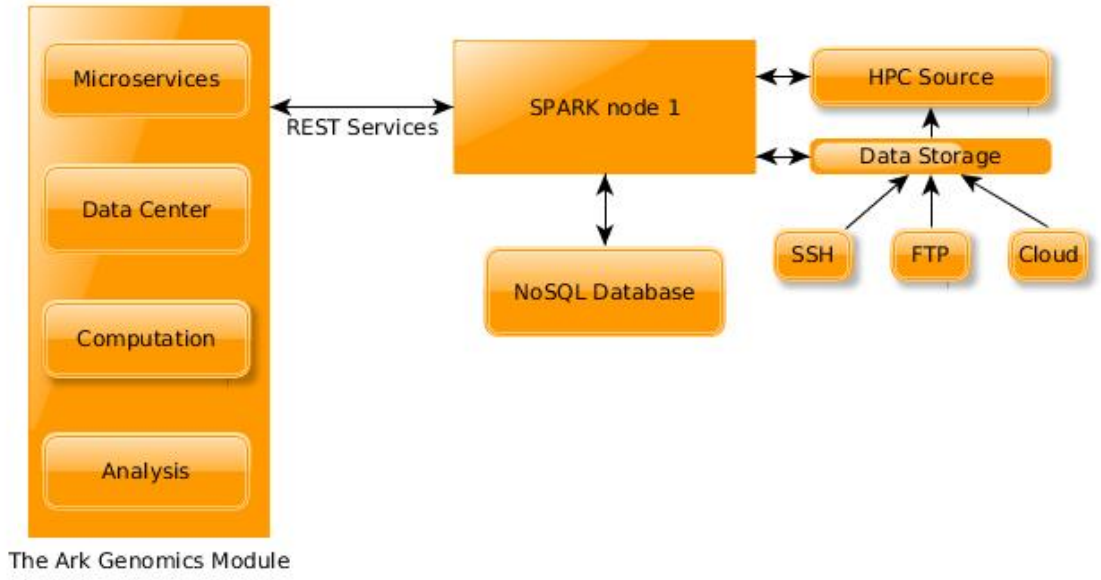


Figure 5.4: The high-level architecture diagram represents the Ark Genomics module, and external SPARK service centres have been connected according to microservice software architecture.

on this service.

The GWAS-related data sources and computing facilities are embedded with the secure access restriction to prevent unauthorised access. These security constraints are unique to data storage and computing facilities (e.g. SFTP and SSH). Therefore, secure credentials are generated for individual researchers or study-specific keys. SPARK's microservice design has eliminated the need for generating individual access keys for researchers and enabled a SPARK node access key to be secured within the deployment server. User access restrictions are managed by the Ark Lightweight Directory Access Protocol (LDAP) secure login protocols and provide encapsulated security mechanism restricted to the SPARK node. The microservice design enables multi-stage security restrictions by the number of security levels implemented by the secure Ark user login, secure web services to connect individual SPARK nodes, and facility-specific access restrictions configured in the SPARK node. Security restrictions are not visible to the researchers who carry out their

research activities within the limitations of the existing Ark authorisation levels.

5.4.1 Micro services

Based on the microservice SPARK design architecture, GWAS researchers do not need to own a HPC computing facility or study the operating manuals of the computing facility. The facility-specific operating instructions are embedded in the SPARK node implementation which interacts with it. The researcher has to be an existing Ark study user who is authorised to use an existing SPARK node and eligible to access the operational procedures for any computing facility represented by the SPARK node. The SPARK microservice design enables the decentralised governing mechanism to conduct the analytical procedures via a common web interface in HPC facilities.

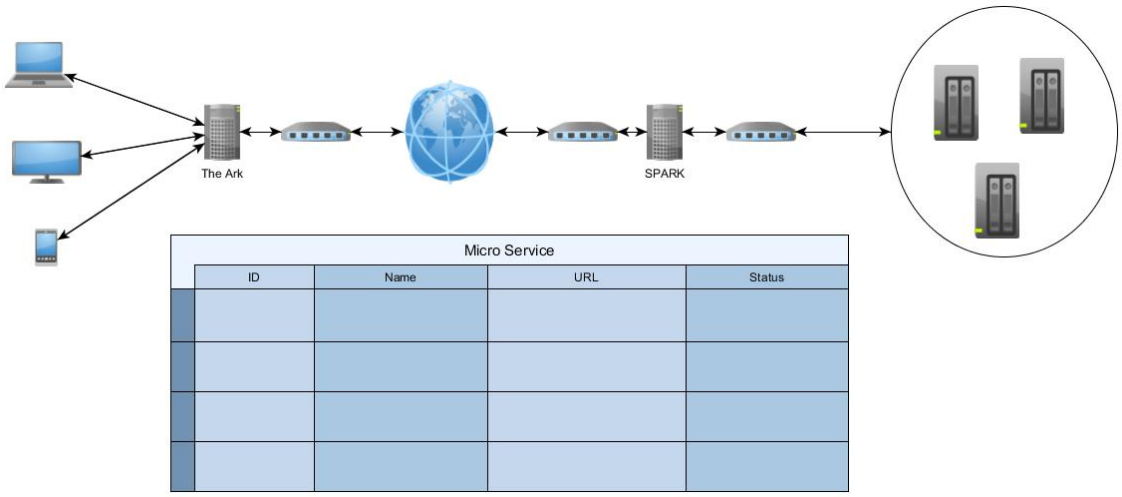


Figure 5.5: The micro service interaction with distant SPARK nodes.

The micro service tab in the genomics module declares the identities of the service end points. Here the service locations are specified by a web user interface, and researchers can point their analysis to these micro services to execute and observe the experiment results. The service initialisation is limited to the Ark super

users to prevent security malpractices. The micro service declaration is defined by a unique identifier, service URL and description parameters. The availability of the service is indicated by a status flag embedded in the micro service search list with a test button to check the availability at any time (see Figure 5.6). These micro service entries declare the availability of services by their public URL and give the opportunity to researchers to select specific services to start an analysis. This has provided the secure service end point definer using a simple web interface. Each service URL is mapped to a unique identifier to be referenced in other GWAS functions (see Sections 5.4.2, 5.4.3, and 5.4.4)

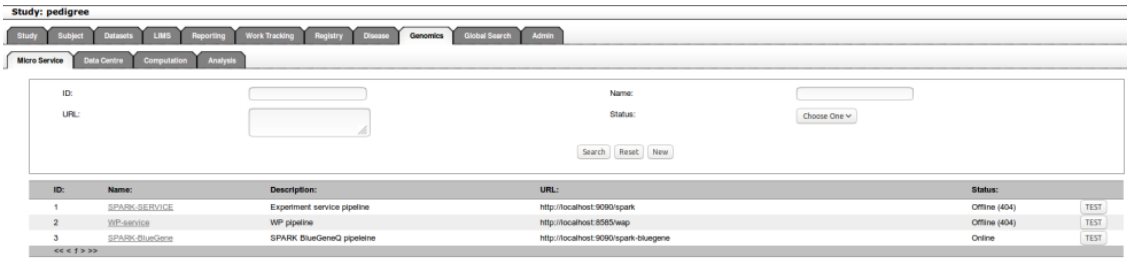


Figure 5.6: The web user interface design for the Ark Genomics module micro service tab.

5.4.2 Genomic data management

The microservice architecture is based on the fundamentals of decentralised data storage and governance (see Section 5.3.4). The SPARK nodes deployed to the analysis infrastructure has enabled the opportunity to manage facility-specific data storage facilities. Multiple data storages can be attached to a single SPARK node supported by a number of data access protocols (e.g. SSH and SFTP). In addition to data storage, SPARK design enables researchers to web-based traverse and query the data sources which are usually manually handled by third party tools. There are individual file traverse mechanisms specific to the access protocols but simplified to common web interface by the SPARK microservice-based design. Querying

is enabled via online and offline mechanisms implemented to enable efficient storage and fast extraction of information from GWAS datasets. As mentioned earlier, microservice SPARK architecture has enabled the decentralised GWAS data management mechanism to ease the overhead of securely managing the research data. Chapter 6 will further elaborate the genomic data management mechanism specified in the SPARK design.

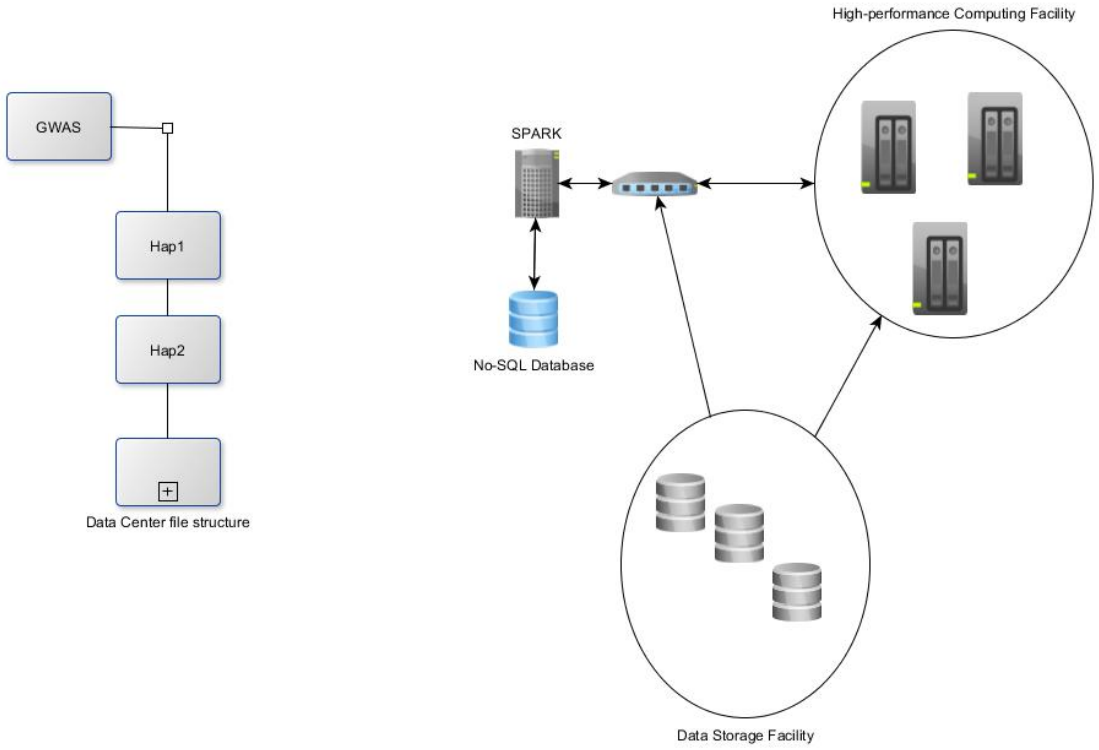


Figure 5.7: Data centre has been chosen based on access type specified in the micro service and researchers can select a file or a directory and declare a data source.

The Ark Genomics module data centre tab is designed to access and query the datasets attached to data centres. This tab allows researchers to select a micro service and identify its available data centres. At this point, users can traverse through the data centre file system via a set of web controllers (links and buttons) by following the data centre root folder structure. The file search is based via the

web controllers and eliminates the possibility of the researcher, too familiar with the expensive command, to traverse the original file system. When selecting a directory or a file, the researcher can declare a data source that will be later used to run the analysis. The data source details include meta information (chip information, the number of participants, SNP variables) about the GWAS file or folder. In addition to choosing a data source, researchers can export the data source to the NoSQL database. Then, genomic data management is capable of using a predefined set of queries to extract the results. These queries enable users to select specific SNP set based on a subject, SNP identity, or group of SNP identities and re-use the results in an analysis (see Figure 5.8).

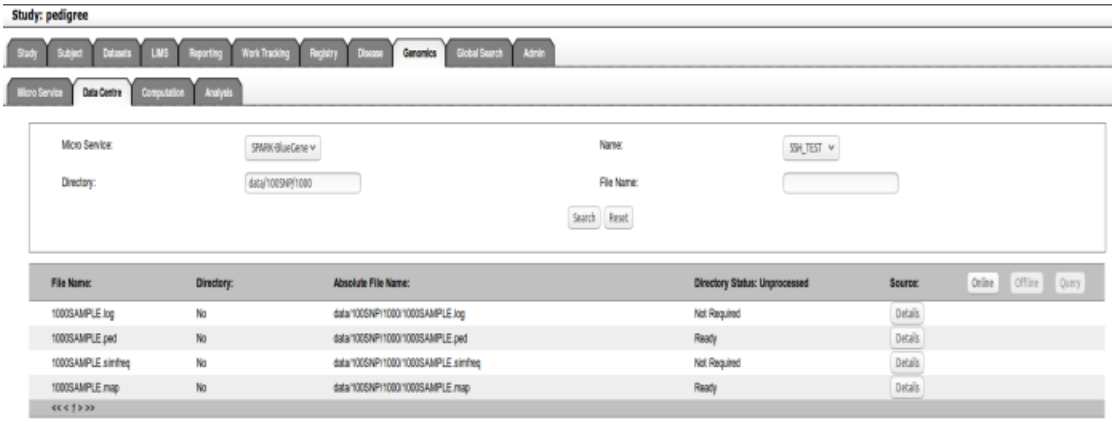


Figure 5.8: Data Centre navigation tab with a selected data source.

5.4.3 Computational packages

The SPARK microservice design has a web-based repository to store analytical methods as computational packages. These can be shared by study users who can access the Ark Genomics module. One common attribute of the computational packages is implementation specific to the execution infrastructure. The SPARK microservice architecture provides the infrastructure-specific execution environment for the analysis. Therefore, the stored computational packages in the Ark can be

deployed to the SPARK node that is suitable for the execution specification. The SPARK computational implementation has enabled researchers to share multiple computational packages suitable for different execution environments via a single web user interface (see Figure 5.9).

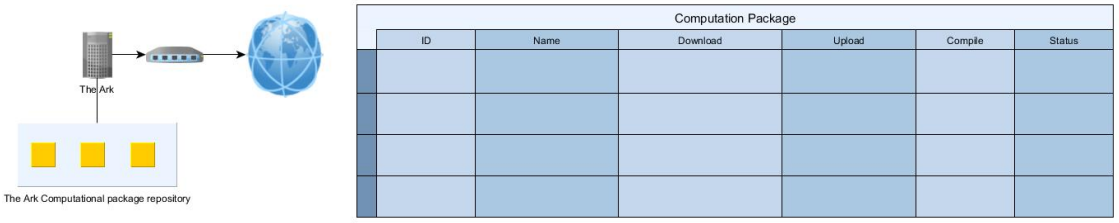


Figure 5.9: The available computational package list and existing computational package with its current status.

The structure of the computational package is predefined and unique across the analysis methods (see Figure 5.10). As a computational packaging mechanism, it has followed the ZIP archive structure to compress the existence of package content. Creating a package archive helps to best use the upload time and reduce the physical disk space required to store the packages. The computational package archive consists of two major parts. The first is the program directory, which contains the source code or binaries of the package. The second is the spark.info file, which contains the execution instructions to build, deploy and execute the package in the selected SPARK node. The mode property indicates the automatic or manual execution state of program files, where automatic implies that program files are in the execution ready state, while manual means program files need to be compiled to become execution ready. Make type and make parameters indicate compile configurations of the program when it is built by the SPARK node. The output parameter shows the outcome of the build process. The run property describes the execution instructions of the computational package when used for analysis (e.g. which cluster and how many high-performance cores are to be used for analysis). The result parameter declares the outcome of specific analysis used by given computational package.

When a researcher creates a computational package in the Ark Genomic Module, the system adds a record to the Ark tables and stores the computational package in the Ark file repository. After that, the researcher can decide which SPARK node is ready to deploy the computational package and upload the package to the SPARK node indicated by unique micro service (see Section 5.4.1). The uploading process is carried out via the secured REST binary web service, the package archive is transferred to the SPARK node deployment environment and the archive is extracted in the deployment environment. The package status is marked as uploaded and can be compiled in the execution environment using the instructions specified in the spark.info file. The compile process completed package status is flagged by the Processed for the Analysis. The computation package implementation mechanism has separated the deployment process from the Ark web container and distributed the responsibility to distant SPARK nodes to carry out in a fail-safe and secure manner specific to the environment. Researchers avoid being exposed to complex computational overheads in the package deployment and are limited to a few button clicks. The uploading and compiling processes are monitored by batch processing calls of the Ark and the waiting period of completing the above-mentioned tasks are avoided. Researchers can move to another application task in the Ark rather waiting to complete the process. Any of the computational package activity that has failed (uploading or compiling) will be indicated by the error status flag in the page list.

The Ark Genomics module has a computational tab to manage the computational packages. The computational tab provides a common interface to manage different GWAS analysis techniques package into the zip archives. Using the computational screen, researchers can define a computational package with a unique identity, upload a computational package to an executable environment or compile a package based on the program's source code. The computational screen is capable of declaring a computational package when it is uploaded and ready to execute. The current status of a computational package will be indicated on the screen by

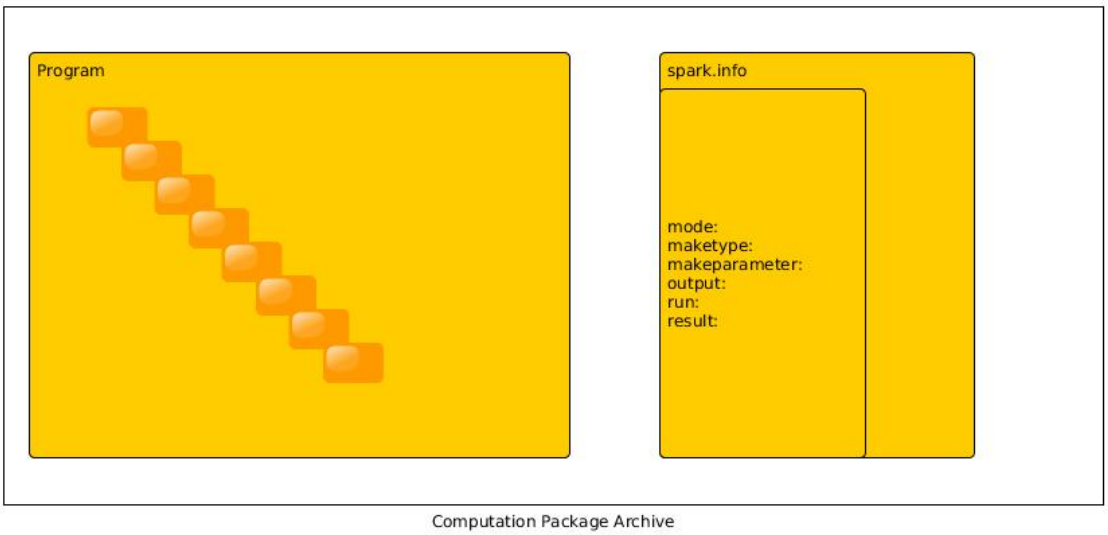


Figure 5.10: SPARK computational archive and its meta information reside in spark.info to declare interoperable parameters.

the status flag in the computational package list in the search screen package list.

The search screen allows researchers to select a computational package by searching wildcard supported name and identity search. Researchers can download computational packages by clicking the download button. Users can delete computational packages via the web interface as an ordinary Ark record rather than manually logging into the execution environment. Computational package management in the web user interface is bound to predefined validation criteria by required fields (name and micro service identity) and delete checks (avoid deleting processing state packages to avoid synchronisation between the SPARK node and the Ark) (see Figure 5.11).

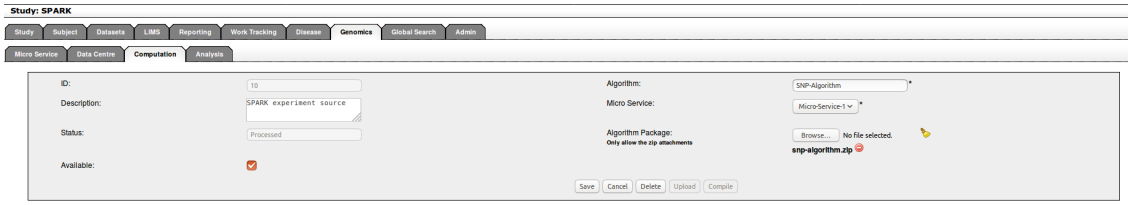


Figure 5.11: A computational detail screen with attached analysis package.

5.4.4 Analysis and results

The outcome of GWAS analysis is crucial for two reasons. The first is the outcome of a specific analysis of a selected data set. The second is using the results as an input to another analysis. The SPARK microservice architecture has enabled analysis-specific results storage so that researchers can examine the results of the completed analysis via a common web user interface and download them for further analysis. Analysis and results design has enabled a web-based result repository for the GWAS analysis and shared results among the Ark study users (see Figure 5.12).

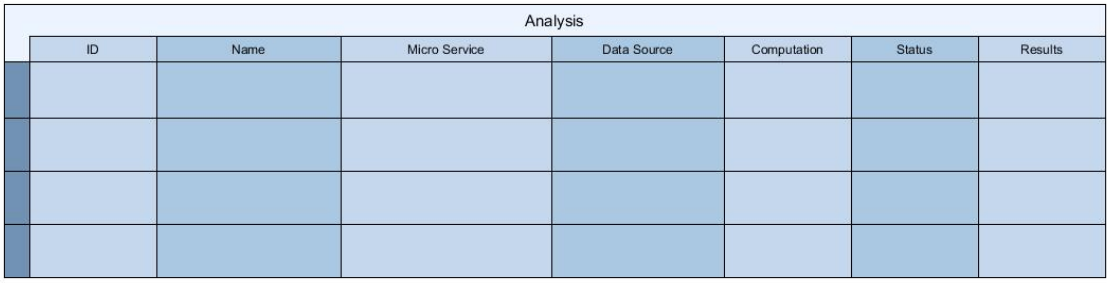


Figure 5.12: The summarised view of the Ark Genomics module analysis section.

The analysis execution is unique to the HPC facility. The execution instructions of an analysis are covered in the computational package section (see Section 5.4.3). Here, the researcher can initiate an analysis in the physical execution environment attached to the SPARK node. When creating an analysis, it is important to choose a deployed computational package and data source for the analysis. For every analysis, a unique directory is created inside the package extraction root directory with unique analysis identity (see Figure 5.13). The selected data source would be copied to the analysis directory when the analysis is started. When the SPARK node receives the execution call, it executes the instructions specified in the spark.info configuration file.

Starting an analysis enables a web service-based synchronisation process to monitor the current status of the analysis and update the Ark analysis entry at specific

time intervals (every 10 s there is a status update call issued to monitor the analysis). When analysis is completed, the status flag indicates the completion of analysis and enables researchers to download the analysis results based on the REST binary call. The analysis results remain in the analysis directory and are available to download until analysis records are accessed via the Ark web interfaces.

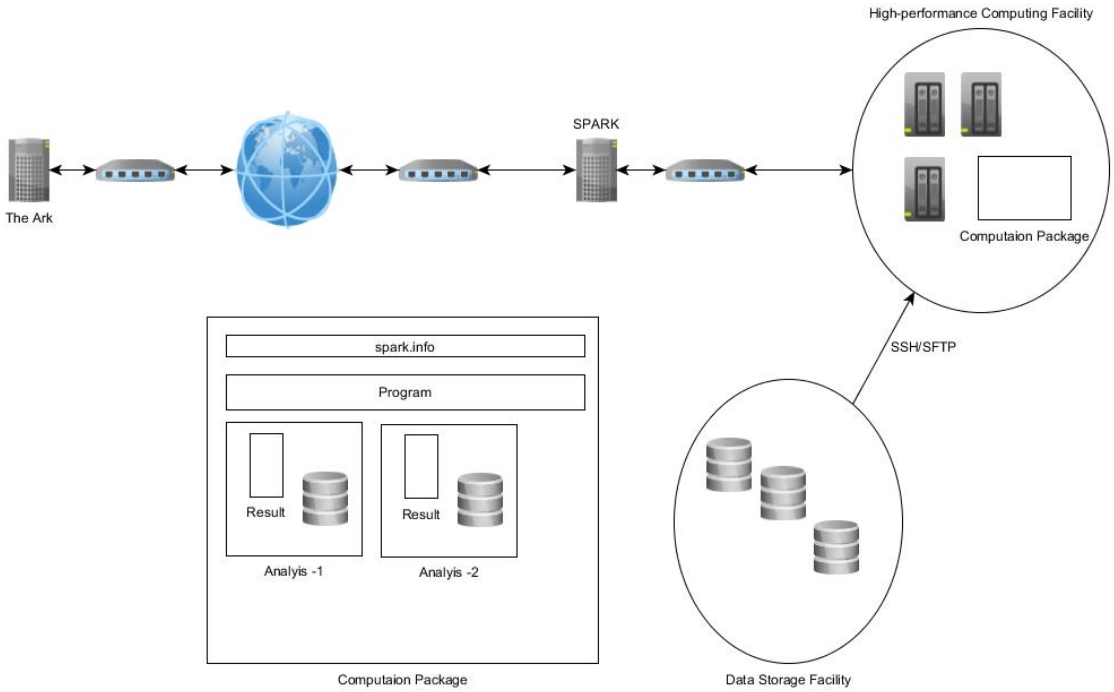


Figure 5.13: The analysis execution and result extraction of the analysis based on computation package and data source.

The Ark Genomics module analysis tab allows the user to execute the computational packages for selected data sources. This process has simplified the much complicated and time seeking GWAS analysis setup [199] into a single web page. Selecting a micro service gives the researchers the opportunity to select the computational packages and data sources available with the service and execute an analysis by clicking the analysis execute button. By clicking the execute button the executed analysis is completed and results can be downloaded via the web interface (see Figure 5.14).

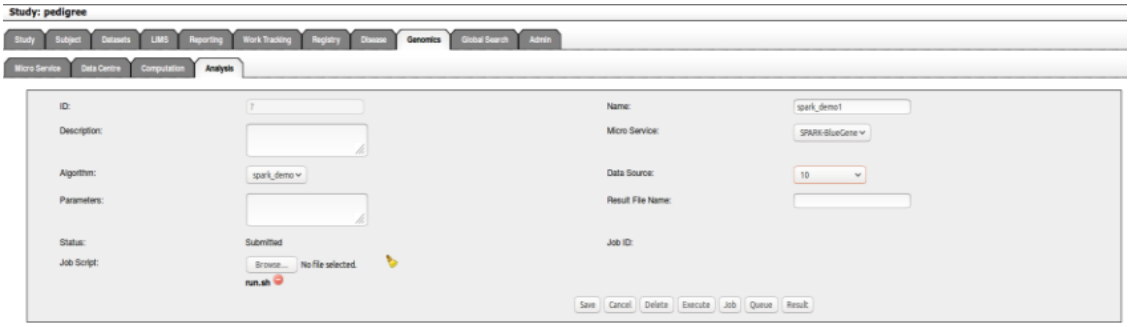


Figure 5.14: The Ark Genomics module analysis web user interface.

The analysis definition is bound to a single web user interface that has a set of validations and restrictions to enable a reliable analysis. Prior to execute an analysis, it is required selecting a micro service, a data source and a computational package. Results download is enabled only for a completed analysis and researchers cannot download the error status analysis results. The analysis search screen enables researchers to traverse through the existing analysis definitions by using their identity, name, bound micro service and status.

Amalgamated with the micro services, data centres, computation and analysis sections, the Ark Genomics module is capable of managing a full-scale GWAS analysis using the SPARK nodes as a single analysis or chained analysis using a set of shared resources (data sources and analysis results). This leads the general software design to support the GWAS management system which is based on the above mentioned functional components. An extendable software architecture that supports distributed computational facilities has led to successful implementation of the SPARK system. It has been a significant challenge to design this software module to integrate with traditional modular software architecture. However, it must also acknowledge the principles of software design best practices to develop the SPARK system and integrate it with the Ark system. Importantly, this approach has symbolised the successful integration of modern data management and analysis techniques to an existing data management platform.

The SPARK microservice design architecture has enabled a wide variety of study

requirements to be captured in GWAS. The next section will discuss the software architectural best practices that are satisfied by choosing the microservice architecture for GWAS management in a study-based data management environment.

5.5 Technical evaluation

The SPARK system has been designed based on the microservice software architecture and the design decision was followed by software architectural best practices. The technical evaluation section will further discuss the software design best practices and their implications for the SPARK design.

5.5.1 Security

Security is a key aspect of a software system. It is important to enable a secure environment to protect the information residing inside the system and provide higher confidence to software users about the security of their dataset. A secure software design includes confidentiality, integrity and availability [200]. Confidentiality ensures the protection of sensitive data residing inside the system from unauthorised access. Integrity guarantees that existing data cannot be modified by unauthorised parties and also maintains a consistent data flow throughout the data lifecycle. Availability enables information to be accessed by authorised users at any given time without delays.

Based on the security best practices, it is important to design a mechanism to react when there is a security breach and alert the relevant parties. A similar alert scenario includes detecting a security breach, restricting a security violation, reacting to the consequences of a security violation and recovering from the situation [201]. Detecting a security breach consists of identifying an abnormality in the general system functionality and data patterns. Furthermore, a common implication of a security breach can be identified by denial of service, message integrity

or a delay in a message flow. Preventative measures include user identification to authorise and authenticate users, introducing user access restrictions, encrypting data elements, separating business entities representing the actual data elements and changing the default security settings (passwords, secure communication) over time. In a security breach, the system should be capable of partially shutting down a selected set of services to restrict the unauthorised access and inform the system administrators about the attack. In the recovery phase, the system is required to maintain the audit history and re-establish the existing service and functionality. Therefore, it is quite important to consider these security implementations in the SPARK design.

Secure storage infrastructure for GWAS datasets, computation methods and analysis results

Medical data, including the GWAS data, provides an example where data need to be handled with care. GWAS datasets reside in the SPARK system related to the personal information about the study participants including family relationships, bio specimen information and analysis results, questionnaire responses, and relevant researcher information. Datasets that contain personal information are often bound by strict legal and ethical considerations. The information that resides in those datasets are crucial for medical researchers, but illegitimate access by unauthorised parties could cause disruption to society. In particular, a leak of the medical information can be catastrophic to the medical research community as it might cause social distraction and discouragement from medical study participation. Moreover, such problems can create legal issues for involved research groups and institutes and discourage society from investing in medical research. Therefore, strict security measures are essential for medical data management systems and the Ark has a specifically designed user role-based data protection mechanism to secure the phenotypic datasets (see Section 3.12). Therefore, the data-related activities for users has been restricted (Insert, Edit, Delete and Read) based on their role

(Administrator, Data manager and Read-only user).

SPARK is designed to manage large amounts of genetic information of medical study participants generated from genetic sequence chips. Genetic datasets can be distributed across multiple data centres (see Section 5.4.2) with distinctive security mechanisms (Secure Access protocols, User role-based file and directory access restrictions, and User authentication processes) to protect the datasets and avoid exposing all the data centres in a single access violation scenario where security breach in a single data centre will not affect the other centres. Other than the genetic data, SPARK stores the implementations of genetic analysis algorithms as computational packages (see Section 5.4.3). The stored computational packages contain some of the unpublished analysis techniques which are required to be kept in a safe location. This includes safety storage of the source code, a safe built environment for the algorithm source and a secure execution environment for the algorithm provided. The analysis results are stored in a secure environment and compliant to the existing Ark security controls. Also, data centre, computation and analysis interactions are bound to the existing Ark user roles and authorisations (read-only user, data administrator and study administrator).

Microservice-based security infrastructure

In addition to securing existing datasets, SPARK needs to secure the access points of the HPC infrastructure and data storage facilities. The access points represent the login credentials and secure communication protocols to interact with a computing facility as a gateway. These infrastructures commonly have high-security standards (see Section A) to prevent unauthorised access and SPARK has to implement a secure communication mechanism with the HPC and data storage facilities to avoid possible access violations.

As discussed in the previous section (See Section 5.4), the SPARK system has to be compliant with its original design goals. The microservice-based (see Sec-

tion 5.4.1) SPARK services design enables secure distributed resource management. The distributed resources included the specialised HPC facilities and data storage facilities specialised to manage big data sets. As discussed in the Ark chapter (see Chapter 3), the LDAP-based user authentication and user group access authorisation mechanisms are utilised to implement the secure data environment for the Ark. The SPARK system has been developed as a module of the Ark system (Ark Genomics module), therefore SPARK users are Ark users and implicitly obtain existing security infrastructure from the Ark.

The existing Ark secure infrastructure has given protection for data residing in the Ark Genomics module which works as the common web interface for distributed SPARK modules. There researchers interact with SPARK nodes via the Ark web interfaces based on existing security constraints. Furthermore, it enables user authentication to the system, module permission to the user groups, and secure infrastructure for the data residing in the genomics module (Meta data of the data sources, Computation packages and Analysis and results).

Another security design aspect of the SPARK system is the SPARK execution nodes which reside outside the reaches of the Ark secure infrastructure. SPARK nodes are operated independently of the microservice architecture, and the Ark secure implementation maintains secure communication with those nodes. The secure communication is enabled via the secure REST protocols and data are transferred as encrypted messages.

Remote SPARK nodes have their own security infrastructure and application level security architecture. Therefore, the Ark Genomics module communicates with appropriate services using secure REST protocols similar to the approach followed during a secure web page access. This enables the Ark Genomic Module to send and receive information with remote SPARK nodes using a secure communication channel. This approach will preserve the existing security best practices (Secure Access protocols, Login mechanisms) of the remote SPARK nodes and separate the security architecture from the Ark infrastructure to avoid total secu-

rity malfunction in an emergency in a SPARK node. Similarly, such design allows selected SPARK services to be shut down during a security breach.

In addition to the REST security protocols, SPARK micro services inherit security infrastructure best practices (system-based periodical security updates, choosing secure access protocols, and educating the secure system administration). High-performance computing and data storage facilities usually enable single access protocols (e.g. SSH, SFTP, and OpenStack). These access protocols would be unique to the facility and protected by the username and password combinations or public and private key control. To satisfy the requirement of acknowledging multiple access protocols, SPARK application nodes are operated as web nodes deployed in a separate web server and attached to the remote facility. Such application deployment and execution architecture has provided additional security for the SPARK system. As illustrated in Figure 5.15, only the web components are available for public access. This creates an extra barrier by adding more security parameters between the web components and infrastructure to prevent unauthorised parties accessing the secure infrastructure facilities.

The SPARK system security exceeds the security of a day-to-day form-based web applications where a web application is usually implemented based on a single security architecture login with password-protected access and session management. Also, SPARK security has data storage and services which are separate from web access to add an additional security layer. The Ark has followed the existing security best practices to provide a secure storage facility for biomedical datasets. When revisiting the microservice design, SPARK implementation has enhanced the existing Ark security best practices and implemented a tailored security mechanism for the independent micro services. This enables secure implementation per micro service interfaced with a single web front end via secure REST protocols.

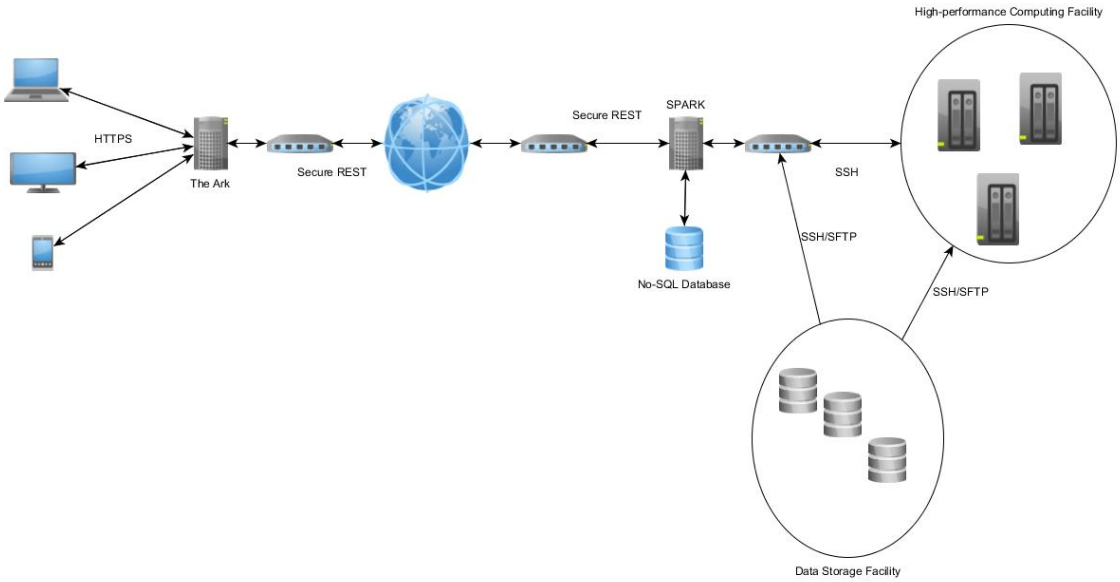


Figure 5.15: SPARK secure architecture with its security best practices to enhance the information related to the system.

5.5.2 Availability

The availability of a software system refers to how frequently the system is available to its users. This includes the availability of individual functionality at any given time, a fail-safe mechanism to handle exceptions and a recovery mechanism to overcome a failure [202]. Mission critical systems need to be available to all users all the time and need to be responsive to the expected use cases of the system. Therefore, systems need to satisfy a set of properties to indicate the high availability of the system. To enhance the system availability for the users, it has to be developed to indicate the known set of functional workflows, clearly identify the data model entities, monitor the mechanism to check the system status and have predefined responses for a failure.

Another important attribute of system availability is how to handle an exception scenario. This is called the fail-safe mechanism of a system. When a system is implemented according to a fail-safe principle, there has to be a set of protocols to handle a failure so that there is minimal impact on the other system

components [203]. A good software system has a monitoring mechanism to assess the status of system components and report any exceptions. Systems have implementation and deployment guidelines to avoid a total system halt when a failure is detected in a single component. Implementing a fail-safe mechanism helps to maintain a stable system, increases the usability of the system and enhances the trust of the users.

A recovery mechanism to overcome system problems plays a crucial role in ensuring the availability of the system. This includes clear guidelines to activate and deactivate system components, a predefined monitoring mechanism to validate system status, a dedicated operational unit to handle system deployment and a predefined set of tests to assure the system status after deployment. Operational tactics would reduce the risk of re-establishing a system after a failure. A well-managed system would be able to continue the operations after a failure without losing data or the systems previous credibility status. The SPARK system has been designed and developed based on the concept of maximising system availability. In the next section, best practices being incorporated into SPARK design and developments to maximise the system availability are discussed.

Micro service status-based operational decisions

Based on the factors related to system availability discussed in the previous section, SPARK has its own implementation goals to maximise system availability. SPARK has been designed based on the microservice architecture and the application component which are independent of each other. To preserve system availability, SPARK has implemented a pinging mechanism to check the status of its associated micro services (see Section 5.4.1). A status flag shows the availability of a service.

According to the status of a micro service, SPARK system enables or disables the required functionality at the service end point. This has minimised user frustration

over unavailable services. Failure of a single micro service does not affect the total system availability of SPARK. The Ark Genomics module registers micro services and then calls on their data centre functionality, computational package deployment on them and executing analysis on selected micro services. Therefore, system availability is largely constant to the Ark system availability rather than an individual micro service.

Service availability-based data and analytical operations

There is an inbuilt validation mechanism to avoid using unavailable micro services in the data centre, computation and analysis sections (See Sections 5.4.2, 5.4.3, and 5.4.4). For example, if a service is down, users cannot access the data centres related to the service. Computational and analysis implementation consists of the status flag to indicate their execution status. If execution fails then the analysis and computation entities are flagged with an error status.

Recovery from exception status is another aspect of the SPARK availability criteria. Micro service administrators can shut down a service node without interfering with the other services and the total Ark system usability. Any failed analysis can be reinitiated in the SPARK system after service status has been changed to available. This approach minimises the functional contradiction between the data centre, computational and analysis workflows. Users can duplicate a computation or analysis on a selected micro service. This availability increases the system's credibility among the SPARK users and enhances the system's availability without infiltrating the service status. The system recoverability is also increased by individual service dedication per facility and changes would be applied per service rather than the Ark system wide fix.

5.5.3 Interoperability

A modern software system needs to communicate with many external instances. These external systems operate independently and have been implemented according to their specification. To produce a collaborated software platform with distributed services, systems should have mechanisms to communicate with each other. The capability of a software system to communicate with external systems and provide collaborated results to users is called the interoperability of a system [204].

There are many advantages via implementing interoperability of a system. These advantages include specialised systems being built to handle specific tasks, attraction of specialised developers to implement specific components, additional security added by distributing application tasks to distant locations and enabling provision for the future changes and enhancements to the system. Modern software systems are bound to the rapid changes and time-to-time system integrations with external systems based on business collaborations and mergers.

Communication is a key part of the interoperability of a software system. Web-based software systems mainly use two types of software protocols to communicate with each other. These are the web service Simple Object Access Protocol (SOAP) and REST. SOAP web services interpret information based on an XML format and can interpret business logic using the Business Process Execution Language and handle transactions using transaction management web service implementation specifications, Universal Description Discovery and Integration Language guides to discover and publish a web service. Reliability is not an inherited attribute to the SOAP-based web service [205]. Therefore, applications that use SOAP web services need to follow the SOAP web service API inherited from their programming language.

The REST protocol was introduced to the application developments after observing the complexity and limitations of the web service SOAP protocol. REST web services run on top of the hypertext transfer protocol and are bound to the hypertext transfer protocol constraints of GET, POST, PUT and DELETE. REST

services are not limited to the XML format and can use any format supported by the hypertext transfer protocol. There are no data type restrictions inherited in XML format. REST services do not have a standard format and can be interpreted by any client application supporting the hypertext transfer protocol. Therefore, any web browser can easily interpret a REST message.

Modern web applications tend to implement either SOAP or RESTful web services to interact with external software systems. Especially in the microservice design-based software systems, web services managed the individual software components and represented the results. Therefore, the SPARK system based on RESTful web services were interoperable with its services. In the next section, SPARK interoperable qualities and the implementation best practices adapted to design are discussed.

Micro service interoperable with multiple high-performance computing sources

As an interoperable instance, the Ark Genomic Module is designed to interact with a number of micro services (see Section 5.4.1) available from distant SPARK nodes. Registering the number of SPARK micro services through the Ark Genomics module defines the initial interoperable constraint to the SPARK design.

Based on the micro service availability, SPARK data centre, computation and analysis functionality in the Ark Genomics module is operated. Therefore, functionality implemented to manage a data centre would be unique to the micro service derived from a SPARK node. Similarly, installing and compiling computational algorithms and executing an analysis is unique to the individual SPARK node supported infrastructure. This has satisfied important interoperability properties of tailor interface and orchestration.

Interoperable computation packages for different execution environments

SPARK nodes are developed based on a common API to define their web services. This has been called the tailor interface of an interoperable system. Therefore, implementation of a method body bound to a web service call could have different meanings but service name, input arguments and return parameters are common for all SPARK micro services (see Section 5.4.1). The computational packages have included execution parameters and commands unique to a HPC facility (see Section 5.4.3). The orchestration is another important aspect of an interoperable system. Therefore, SPARK computational packages included a meta information file to define the executable parameters unique to each HPC facility.

This type of meta information file orchestrates the interoperability of package execution and analysis is conducted in different computing facilities without implicitly defining the execution instructions per analysis. Therefore, Ark Genomics module can interact with multiple SPARK nodes bound to different HPC sources without modifying or adjusting its internal implementation. Orchestration of functionality enables the SPARK system to interoperate with multiple computer facilities without exposing their internal implementation. This helps researchers avoid the burden of exploring the technical specifications of HPC facilities.

Therefore, interoperability is a key design goal of the SPARK system, and this was achieved based on the microservice architecture and computational packaging format. Considering the interoperability as a key design goal, the system is capable of orchestrating the analysis on different high computational facilities and the tailor interface has contributed to the uniqueness of the services operated, based on independent micro services.

Interoperable data centres to management and executing the GWAS data search queries

The SPARK data centres (see Section 5.4.2) operate independently based on the relevant micro service. Therefore, data source navigation and querying are unique for the service implementation. The Ark Genomic Module web interface provides a same functional user interface for available data centres. The SPARK API provides a unique set of web requests and responses for supporting web interface calls.

For example, there can be multiple data centres attached to a micro service which has no similar file management approaches. There can be SSH, SFTP or OpenStack protocol-based data navigation mechanism for each data centre. However, the Ark Genomic Module data centre (see Section 5.4.2) provides a similar web interface for each access protocol, and an underlying navigation mechanism is implemented in the SPARK node implementation unique to the data centres.

5.5.4 Modifiability

Changes are inevitable for any software system. Implementing a change for a software system must take into account the definition of the change, change acceptability, implementation and cost effectiveness [206].

When accepting a modification to the system, designers must evaluate the change and check the change's feasibility to accept a software system implementation process. Accepted by the software designers for a change indicates this modification is essential to maintain the capabilities of a software system and provide a significant advantage for its users. Generally, the implementation cost of the change will be covered by the system performance and user satisfaction achieved from the change.

Coupling and cohesion plays a significant role in the modifiability of a software system [207]. Coupling refers to the interdependency between the software modules of a system. To process a functionality in a less coupling system, modules work

independently and have minimum dependability with other modules to deliver the result. Cohesion is the interoperable functionality in a module itself, and how much interoperable function would be required to deliver a result will be measured by cohesion. Systems with high cohesion and less coupling tend to have modifications. These types of situations are highly effective in conducting modifications by having minimum downtime, and therefore less testing is required on system modifications.

Micro service modifiability options for SPARK design

The SPARK design was highly vulnerable to the modifications due to the nature of GWAS research and the supporting infrastructure. Software systems that support this type of work should be highly scalable and open to changes. To achieve these goals, systems are designed using modular architecture and micro services (see Section 5.4).

In the Ark Genomics module and SPARK nodes, modifications can be applied in two different places. The web-based controlling structures and user interface modifications applied to the Ark Genomics module. As discussed in the Ark chapter (see Chapter 3), the genomics module was developed as a pluggable software module for the Ark system and modifications can be carried out without interfering with the main system functionality.

The infrastructure-related changes were made according to the specific SPARK node implementation. This allows SPARK node owners to carry out their modifications without interfering with other SPARK nodes. The Ark Genomics module operates based on micro service availability, and during the modification period, developers can keep the external SPARK nodes in an offline state. Therefore, the genomics module will omit to call offline micro services and operations are carried out based on only the online micro services .

Independent micro services allow designers to adapt SPARK node-specific modifications in required places. This has created implicit loose coupling between the

SPARK nodes and given freedom to the individual owners to manage their SPARK instance. The Ark Genomics module was carefully designed based on proven component standards of the existing Ark system and keeping close cohesion between the components of the Ark Genomics module.

The optimum modifiability implementation for computation packages

In the computation section the packaging format for the analysis methods is stored as archives (see Section 5.4.3). This packaging format is specific to the executive environment rather than the Ark Genomics module. Therefore, developers do not need to change the computation and analysis modules to match the execution environment.

According to the SPARK specifications and design, the system has implemented maximising the modifiability of the computational packages and encourages developers to maintain the optimum modifiability in their individual analysing methods. Adding more computation packages would increase the modifiability of analysing techniques suitable for different GWAS data formats and executed in multiple HPC environments.

5.5.5 Performance

Good software systems should be designed to enhance their performance by fine-tuning the technical aspects and enhancing the quality of system components. System performance is measured by the time taken to complete an event and the accuracy of the results generated [208]. There are several categories of events including the periodic and stochastic events. Periodic events occur in predictable time intervals and given much needed preparation time to handle these events. Stochastic events also happen according to a predictable time model, and the system can be fine-tuned to handle the situations. Sporadic events are not bound to a time frame and the system is notified after an event has occurred. These events are

hard to predict but can adjust system performance by triggering predefined event listeners when a sporadic event occurs.

The reaction to an event is bound by a set of response attributes. These attributes include the latency of an event, processing deadlines, throughput parameters, response limitations and ignorance. Latency is declared by the duration of time between the event requests, and less latency indicates high system performance. Processing deadlines are declared to define the selected functional aspects which execute when given processes in the place. The throughput parameters are defined to limit the system processes which match up with the available resources. Response limitations are used to control the latency by avoiding unreasonable time lapse between the request and response events. Ignorance is defined in the system to avoid handling unacceptable situations which exceed the system capacity.

Based on the performance attributes of event requests and responses, a set of design guidelines were introduced to the software design. Software design mainly considers controlling the demands for shared resources. Controlling the resource demand includes managing the event rate per resource, limiting event responses generated from the resource, prioritising the events per resource by introducing queue mechanism, reducing overheads by optimising the coding samples, binding execution time by delegating the responsibilities and increasing resource efficiency by third party tools.

To manage the resources, the following tactics have been introduced to the system design. Adding more system resources will increase the system capabilities like more memory to increase the memory heap and additional storage resources to increase the storage capacities. Adapting concurrency at the programming level helps to manage the resources efficiently in multithreading environments. Distributing the computing and storage capabilities reduces the resource overhead. Implementing queue systems and schedules will enable asynchronous execution of processes that are unbound to the request and response cycles derived from the events.

SPARK has been implemented based on optimising the performance factor of

a GWAS management system. The SPARK design has adopted standard performance enhancing tactics to utilise its functionality.

SPARK performance optimisation attributed to its microservice design

SPARK consists of multiple data stores to manage genomic datasets and HPC systems to conduct analyses. System events generated via the Ark web interface are individually mapped to the genomics module components. A SPARK event consists of a request to pre-programmed functionality and response from the outcome of the event or the present status of the event. SPARK has been designed based on the principles of distributed software design following the microservice architecture. As a design guideline, SPARK design caters to controlling the resource demand rather than managing the resources (see Section 5.4.1). Therefore, the infrastructure and resource capacities are managed by the individual SPARK node administrators.

When optimising the performance of a system with controlling the resource demand the following facts would need to be considered. Latency is a critical issue on the SPARK system due to the large size of the datasets it is expected to manage and the complexity of its computational methods. Therefore, complex and time-intensive event requests have been mapped to the schedulers. These job schedulers will be executed based on the event request and continuously reported back with their current status. Based on the event status, the system will define the output type. This approach will reduce resource requests for the same event and also, reduce the responses per event by limiting the event response only to complete status.

Increase performance of GWAS analysis by distributed computational package and analysis

Multiple micro services reduce the overhead on a single resource. Users can distribute their analysis on multiple services by distributing the responsibilities (see

Section 5.4.4). Given mechanism provides more resource control than dedicating a single resource for an analysis task. Managing the coding source as computational packages will allow users to separate the analysis methods from the main system development source. These computation packages can be tested and executed in different micro services without affecting all of the micro services. Every computation package consists of a spark.info file to define the execution meta information unique to the event and can define analysis execution properties by providing the external script and output type.

SPARK has enabled using third party tools to optimise the operations in individual micro services. Researchers can use the execution code as computation packages to be stored and executed in the resource facility. This will reduce development time, compiling time of the source code and testing time for the analysis. Viewing the current status of the facility via the web interface provides better understanding of the execution environment for the users. Individual micro services provide a fail-safe execution environment to researchers by limiting execution to an external micro service rather than the main Ark body.

5.5.6 Testability

Testability is a crucial design feature of a successful software system. The reliability of a system and user confidence in system functionality significantly increases by the testability of a design. When developing a test plan for the design, architects will give special attention to testing options, including observations based on system state changes and reducing the complexity of system components [209].

Observations based on system state changes can be explained by explicitly changing the system status and observing the results. The system can be evaluated for positive events and negative events. Positive events consist of the usual system events with positive inputs, and negative events consist of invalid inputs for the same scenarios. To achieve these objectives system designers consider following

best practices.

Therefore, the specialised programming interfaces will be developed to define the general test scenarios for the system. These interfaces guide developers to create their test cases according to a template. Functional implementations are checked according to a template to verify their reliability. Recording and playback events are used to recreate events when finding an issue or exception in test scenarios. These playbacks are used in the development stages to recreate the issue and verify the fixes. Maintaining a state machine for the objects is good practice in software development. Here, each object is bound to a specific state in its life cycle. This helps to implement test scenarios based on the state of functional objects.

Abstract data sources are used to define test scenarios based on the data types. These test cases would point the system to various data sources which have similar meta information in its real-time database. Sandboxing is an approach to isolate a system from a real world execution environment. A system is deployed in an environment, most probably inside a virtual machine with similar infrastructure and network traffic. The objective is to verify the system capabilities on a real world execution environment before opening the system to users. Assertions are another set of test cases defined in the code base to detect the known exceptions. These assertions will be triggered if system reached these predefined states.

In addition, controlling and observing the system states in testing and dealing with complexity is considered an important role in test architecture. Eliminating complex implementations and delegating responsibilities in small system components will increase the reliability of system testing. Test cases developed to verify the small components will verify the system state at the base level rather than covering complex situations. This approach helps to test cyclic dependencies and individual components. Completing the test matrix for individual components will increase the system reliability by testing the deeply driven functional components. A system with high modifiability, less coupling and more cohesion inherently includes less system complexity. These systems can be tested more accurately and

thoroughly than a complex system design can be tested.

Testability of the SPARK system design

The SPARK design is compliant with architectural best practices. Therefore, the system design is compliant with state evaluation of the functional workflows. The SPARK implementation is bound to a set of interfaces that specify the functionality of the system. The test plan consists of positive and negative test cases that address each method specified in the functional interfaces.

Datasets have been generated for the test scenarios and will be discussed in the data management chapter (Chapter 6). The test cases and datasets are compliant with the existing database design and match with the database's meta-properties. Sandbox testing is inherently addressed via the SPARK architecture. Due to the microservice design, the system can accommodate test services as equal to an production-based SPARK services. Before exposing a real-time micro service (see Section 5.4.1) to the SPARK interface, there is an opportunity to deal with its mock implementation. Tests can be carried out with the mock service to verify its reliability and accuracy.

The SPARK system has been designed and implemented based on satisfying the design best practices of high cohesion, less coupling between components and optimising the modifiability. Every component can be bound to test cases to verify the accuracy. This helps to verify the functionality of system components and declare statement on the system functions as a whole.

5.5.7 Usability

Usability is an important aspect of software design. Important usability features include easily learning the system functionality, efficiently interacting with the system, minimising the impact of system errors, satisfying the user perspectives, and maximising the users' satisfaction and efficiency. Ultimately, the usability of a sys-

tem is bound to the objectives of the system and how the system has reflected these objectives to the users [210].

Elaborating the features mentioned above would be as follows. Learnability of the system reflects how easy a user can adapt to the system and understand the functionality to complete a task. This includes help options and a logical user interface design to guide users. Efficiently interacting with the system includes performing multiple tasks simultaneously using the system interfaces and obtaining results in a feasible time frame. There have to be inbuilt validation techniques to evaluate the user input and save the state of an action until process is completed. It is highly valuable to have a mechanism to tolerate system errors in an error.

Software systems are highly vulnerable to system errors and failures. It is difficult to eliminate failure completely and it is better to implement a precaution mechanism to minimise the damage. Therefore, when there is a probable system error, there needs to be an inbuilt mechanism to avoid the user being distracted by the event. A system with high usability design will increase the user confidence in the system and increase the acceptance of the results. More importantly, popularity of the system has a high impact on its usability based on recommendations made by the system user community.

SPARK web user interface design and usability

The SPARK system was developed with usability as a key design goal. The SPARK system has inherited the existing usability features of the Ark system and its web user interface. This includes the basic guidelines and practice for the GWAS management functionality and system interaction features like screen formats, menus and button navigation. To satisfy one of the primary objectives of SPARK design, interfaces were designed to efficiently interact with the HPC resources and big datasets. High-performance computing resources usually communicate via a specialised set of command line commands. SPARK design has encapsulated these

commands by a set of user interfaces under four menu items (micro services (see Section 5.4.1), data centres (see Section 5.4.2), computation (see Section 5.4.3) and analysis (see Section 5.4.4)).

The micro service status and independent service points have minimised the impact of a service failure. Before calling a function, a service check is performed to check availability. The failure of a service only affects the functionality of the service rather than having an impact on the Ark system or other micro services. System error related to a function only impacts on the particular service and does not affect other services. There is always an error handling mechanism at the code level to identify an error in the system and notifies users with meaningful error messages rather than programming exceptions.

User interaction mitigation to enhance the large processing tasks

The SPARK system allows users to execute events (data centre query processing (see Section 5.4.22), computational package initialisation (see Section 5.4.3), and analysis execution and results (see Section 5.4.4)) that take a considerable time to complete. These events normally exceed the time span allowed for a valid web session. To overcome this, SPARK has bound large event requests to schedules and states. When executing, a task is labelled by a state and an interface declares the system functionality based on the event state. For example, if an analysis is started then the state is set to processing and the result extraction button is disabled until the change in state is completed. State monitoring will be handled by the scheduler events triggered at predefined intervals.

The advantages implied by microservice architecture with independent service implementation and execution include labelling the requests with status labels, simplifying the complex operations by a well-structured set of user interfaces and handling the exceptions internally. These increase the system's usability. The usability scenarios of the SPARK design will be further benchmarked in Chapter 7

in a comparison with existing GWAS analysis and management systems.

5.6 Conclusion

This chapter has discussed the software system requirements for GWAS data management and analysis and the software architecture design needed to enable GWAS data management in a biomedical data management system. Therefore, microservice software architecture has followed and developed a web service-based GWAS data management system based on the Ark. As a prominent study-based biomedical data management system, the Ark has been introduced the genomics module interfacing with the distributed SPARK nodes.

The Ark Genomics module consists of four sections to interface with distributed SPARK nodes to manage the GWAS datasets. The micro service section identifies and manages distributed SPARK nodes. The data centre section communicates with the data sources via multiple access controls (SSH and SFTP) and queries the data sources by configuring query parameters. The computation package section stores and initiates the analytical packages which are compliant with multiple HPC platforms. The analysis section manages the execution algorithms for selected data sources and manages their analysis results. Choosing microservice architecture for the SPARK has enabled a unique software solution to manage heterogeneous GWAS datasets independently and has simplified the much complex analytical pipeline design and configuration into a simple set of user interfaces.

The SPARK microservice architecture has been evaluated based on well-known software quality attributes and highlights the selection of the system design against the other software architectural design specifications. The microservice design has provided an advanced security environment for GWAS datasets and analysis, including existing Ark user role-based security authorisation, separating the execution environments by applying multiple security layers, and standalone security implementation native to the independent SPARK node. The availability of the system

is increased by multiple micro service nodes which increases the chance of available resources for a single researcher. Execution of the analysis is not limited to a single SPARK node; it is possible for the researcher to use any of the micro services available at the execution time.

The interoperability of the SPARK architecture is enhanced by the use of the RESTful web service to communicate with individual SPARK nodes. Algorithms implemented for analysis can be stored as multiple versions in the computational packages section. The changes to the SPARK nodes are kept independent of each other by maintaining a unique implementation of data source management, computational package handling and analysis. Therefore, the modifiability of the SPARK design is maximised by isolating the changes occurring in each SPARK node. Performance enhancements are maximised by enabling facility-based hardware to increase processing power and physical disk storage without being limited to the typical web application host server. The testability of the SPARK design is increased by not being limited to single micro service and having the opportunity to initiate multiple service endpoints and compare the results. The usability of such a system is a crucial factor of consideration and the GWAS system was simplified to four components, including micro service, data centre, computation package and analysis with the simple Ark web user interfaces.

This chapter has discussed the software design and implementation approach to support GWAS data management with the maximum use of available computing resources. The enhanced design of the Ark system has provided a case study for genomic data management in the existing phenotypic medical data management systems. This is a much-needed requirement identified by the genomic research community and provides a solid platform for computer scientists to enhance the design to further genomic research, for example full genome sequencing. The SPARK architecture discussion is focused on microservice design to integrate the distributed high-performance sources via web protocols and to reduce the overhead of using computing resources. Another factor which requires discussion is GWAS

data management inside a relational schema-based data management system. The genomic data management chapter will further discuss the GWAS-specific data management design discussed in the data centre section and specialised software design architecture implemented to manage and analyse the GWAS datasets in a non-relational database environment.

A Big Data approach for GWAS data management.

6 Genomic Data Management

6.1 Introduction

This chapter serves to introduce a novel GWAS data management approach based on software design and a non-relational database schema approach to optimise GWAS data management by utilising disk space and operational times. GWAS data operational time utilisation, based on minimising the loading and extraction times, is considered. SPARK GWAS data management design has introduced a mixed model consisting of relational and non-relation data schemas supported by multiple data tiers of a single application node.

Section 6.2 discusses the properties of a genomic dataset and how it categorises GWAS datasets into a special set of genomic data. The common properties of a

genomic dataset and their variations unique to data types are considered, including the GWAS datasets with their meta characteristics and sample-specific properties of size and variation.

Section 6.3 considers how the GWAS datasets are different from ordinary epidemiological datasets. The best fit of the epidemiological dataset inside the relational database schema and long-lasting success of the study-based epidemiological relational data schemas is examined. A discussion of the challenges in the relational schema follows, including the introduction of the GWAS datasets to epidemiological research. Handling GWAS data structures in the relational schema has made a significant difference to study-based epidemiological research datasets by increasing the number of Single Nucleotide Polymorphisms (SNPs) on chips and samples per study. Eventually, the number of datasets will exceed the capacity of relational database systems.

In Section 6.4, typical GWAS data management processes, including Loading, Cleaning, Querying, and Results analysing processes, are discussed. Each process consists of extensive computer processing acquisition and multiple third-party tools are used to support each individual step. PLINK is a standalone GWAS data processing tool which supports these individual steps but also uses multiple steps specified by the predefined GWAS execution steps. PLINK provides GWAS data formats to standardise the datasets hence reducing the preparation time for GWAS processing of multiple scripts. Standalone PLINK implementation has limited its capabilities to a single computer and provides a user-friendly web-based distributed data processing approach for collaborative GWAS researchers.

The relational data management section has outlined a GWAS data management approach via a standard relational data schema. Data is presented via a normalised table schema which avoids data record repetitions but is connected via a set of unique keys specific for each record which resides in the table. Due to long repeating allele information per participant, the GWAS has limited the per column representation of allele information for individual subject records. Relational

database management systems are limited by a maximum number of acceptable columns in a table. Therefore, an alternative table schema with a pivoting technique was introduced and the disk space allocation and loading times were measured for the MySQL relational database management system. Due to the time it takes to load a moderate GWAS dataset and the limited physical disk space, the allocation rate has signalled to search for an alternative solution to manage GWAS datasets.

Non-relational approaches for the GWAS data management is discussed in the next section. Popular non-relational data models, including key-value, columnar, document, and graph models with relevance to GWAS data management, are discussed. Based on the GWAS data representation in each model, the columnar model has been chosen to further investigate as the best possible data model design for the GWAS. The next section considers a novel approach which provides efficient GWAS encoding for a columnar database. An alternative data representation of allele data based on the columnar module is used to avoid inefficient character-based allele encoding. The columnar model has been guided by existing binary data representation of the PLINK data formats but is enhanced to utilise the maximum benefits from a database management system. After conducting experiments based on existing GWAS datasets used in the relational database, the experiment has demonstrated that the advantages of following the columnar-based binary allele encoding model are the significant decrease of disk space acquisition and loading times.

The SPARK design and implementation have introduced data modelling architecture for a study-based epidemiological data management platform. In the final section of this chapter, the existing data architecture implementation in N-tier based software design and its implications to modern GWAS data management is examined. The SPARK architecture is based on microservice architecture design, yet the individual SPARK modules are based on the N-tier architecture at the micro context level. The data tier is responsible for the data-related operations, yet with table data and large processing jobs, GWAS data management is required to per-

form complicated data management activities rather than direct data mapping. To enable such directives, Lambda architecture-based data-tier revamp has been implemented and the advantages and disadvantages of implementation are discussed. The given data architecture can be considered a role model for GWAS data management implementation for Big Data techniques and software design model for similar systems.

6.2 Characteristics of a genomic dataset

Genomic datasets are formed from the outputs of genome sequencing and biological processes. These datasets contain various genetic information types, including SNPs, gene structures, RNA structures, and protein structures. SNP datasets represent the variation of a single nucleotide in a genome and are specific to a certain position of the genome of same species. According to the NIH reference[1], there are 10 million SNPs residing in the human genome. The gene is a specific region in DNA which represents the protein-encoding information. There were two steps in converting gene information into proteins. The first step is transcribing where DNA region is transferred into a similar structure called RNA. These RNA structures are called the messenger RNAs (mRNA) and are later translated into protein structures[2]. The GWAS dataset is a special type of genomic dataset based on study participants' SNP data derived from special sequencing chips (see Section 2.4) and presented in case/control basis. The GWAS genomic dataset represents the participant's identity (id), family id, parental ids, sex, affected status (case/control), and SNP allele sequence for every study participant. It also consists of meta information (SNP id, chromosome Id, genetic distance, and base pair position) related to an individual SNP which represents the allele sequence for each subject in an ordered sequential manner. Given information sets have been analysed to discover

[1] `https://ghr.nlm.nih.gov/primer/genomicresearch/snp`
[2] `https://ghr.nlm.nih.gov/primer/howgeneswork/makingprotein`

the common alleles associated with the cases to identify genomic material which contributes to specific phenotypic cases (see Section 2.4).

Broadly, this type of information is referred to as 'omics data' [211]. Omics datasets are mostly stored in text files or binary archives. A genomic text file consists of alphanumeric characters based on the repetitive patterns. According to the dataset type, these patterns are varied but maintain a logical representation to store the genomic information. One or more files would be required to manage a specific genomic information set. A dataset typically comprises one or more files, where the data in each file may encode as a binary sequence (0s and 1s) or as text.

Similar to the genomic dataset stored in a text file, binary archives represent the compressed version of these information sets (e.g. PLINK (see Section 2.4.2) binaries). The main advantage of storing genomic data in a binary format is that a significant amount of physical disk space is saved. For example, Figure 6.1 represents the text representation of a PLINK-based genomic dataset and its equivalent in the binary format. The text version of the genomic data with ASCII characters have obtained 8 bits per character but the binary version only needs 2 bits. The downside of this approach is reducing the understandability of datasets without using third-party tools or self-made extraction tools. The text and binary representation of genomic datasets raised the question of "Why not use a database to store these datasets and combine it with a more systematic storage system?"

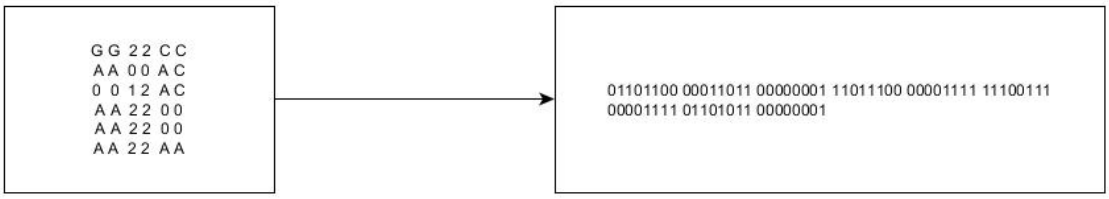


Figure 6.1: A snippet of the PLINK text-based PED file alleles and its binary representation. Sample dataset obtained from PLINK file formats [3].

The option for this approach is to use an available database management system to store the genomic datasets. The popular choice is using the widely available rela-

tional database management system to store the genomic datasets. This approach is relevant and feasible for small genomic datasets which are within the limits of relational database management system capabilities. The small genomic datasets would not exceed the capacity of relational tables with a number of columns and rows. As an example, genomic data consisting of 100 SNPs with 100 populations would make an acceptable scenario, but large GWAS studies consist of 100,000 SNPs with thousands of populations which makes the situation complicated (see Section 6.5). The search queries should be able to execute within a feasible timeline to extract the specific information for a small genomic dataset.

Large genomic datasets have exceeded the capacity of existing database management system capabilities based on their attributes and the initial system design objectives. The existing data management systems are designed to manage the datasets within a known set of limitations, including predefined data models and table size limitations (see Chapter 2). Therefore, it is difficult to accommodate a large multi-attribute dataset which represents multiple definitions for the same data value in different domain perspectives and add complex relationships between the data elements. Even though the storage requirement is satisfied by adding more physical space, heterogeneous genomic datasets have exceeded the scalability and querying capability of a relational database management system.

6.3 How GWAS datasets are different from epidemiological research data

A usual epidemiological research dataset can be explained in the normalised context and fit into the capacity of the standard relational database schema. Further elaborating the capacity of a relational database, including inherent ability to scale the dataset and produce query output within the feasible time limit. When the epidemiological research dataset is stored in a relational data schema, users can access the dataset content by visiting the table structure and column attributes.

As an example, The Ark system database schema is designed to accommodate an epidemiological research dataset (see Chapter 3). Subject datasets related to a study have been stored in the existing data schema and the schema definitions, including the subject-related demographics, questionnaire, laboratory, reporting, and pedigree datasets. The data schema is predefined, and datasets which are going to be stored in the schema have to be matched with the schema definition. Therefore, researchers can extract specific data elements by visiting the data schema via the web interface or the database tables. Also, stored information can be retrieved and modification to existing data element can be achieved within a feasible time limit which exists in a single web session.

For a genomic dataset to be included in the ordinary relational database could cause challenges. The nature of a genomic dataset can be hard to normalise in a relational database schema due to its repetitive nature. For example, taking the PLINK PED/MAP-based dataset (one of the defacto standards of GWAS data which is readable by the PLINK tool) with 1,000 participants with 100,000 SNP alleles would not fit into a single relational database table. Obviously, this would exceed the number of columns accepted by a single table in the relational database schema (most relational database schemas columns per table are <5,000) and time is taken to complete a query execution. Therefore, the most obvious normalised data schema to store such datasets inside a relational database schema would be three tables, i.e. Family, SNP, and Genomic (see Figure 6.3). The Family table represents the individual information with demographic information (family identity, parents, sex, and phenotype), the SNP table represents the SNP meta information (SNP identity, chromosome identity, physical position, and genetic distance), and the genomic table represents individual genomic constraints related to the Family and SNP tables. Their allele columns are the allele representations related to the person and specific SNP. Here, the most important proportion is the genomic dataset which has included only four combinations (AA/Aa/aa/00) (see Table 6.12). This approach will make (m X n, where m = number of alleles and n = number of

subjects) rows inside the Geno table.

The usual GWAS dataset consists of millions of SNPs with information belonging to the multiple persons and only bonds to the four combinations, i.e. AA/Aa/aa/00. Based on the limited number of columns for a table in relational database systems, pivoting techniques where columns are converted to rows must be applied to manage this kind of dataset (see Figure 6.3 for a pivoting example). In Figure 6.3, individual cells of a table have been transformed into rows identified by a unique identity. When applying the pivoting technique, an extra burden of scaling and querying the dataset is introduced. An example scenario would be applying pivoting data to a GWAS study consisting of 1,000 individuals with 100,000 SNPs. It is practically impossible to create row for each person with relevant SNP pairs because most database engine tables are limited to <5,000 columns. Therefore, applying the pivoting technique to a given dataset would create 1,000 columns X 100,000 rows in a relational database table structure. However, increasing the number of rows per table increases the query time and it is hard to scale up the database performance.

Understanding the challenge of storing the genomic datasets inside relational table schema, researchers tend to seek alternative data models. The introduction of NoSQL databases with their alternative data models has made a positive contribution to this challenge. A combination of approaches should be used, including non-relational data models and architectural software design to support storage and analysis of the genomic datasets. The SPARK system was designed to address this challenge by implementing a combined approach to software design and database models to store the GWAS datasets.

6.4 GWAS data management processes

GWAS are massively data-centric and a number of data-related processes will be carried out throughout the study lifetime. These include: (1) loading the datasets

for study data management platforms, (2) quality control process to carry out cleaning and validating the datasets, (3) extract the specific content in the datasets for specific analysis, carry out the analysis on a selected dataset, and (4) analysing the result data output by the experiments. Specialised tools have been developed to carry out these GWAS data management and analysing tasks; PLINK is one such popular tool (see Section 2.4.2).

It is important to understand the loading process of the GWAS datasets in the experiment environment. Unlike specimen-based medical research, GWAS experiments are purely based on the existing datasets [212]. Therefore, datasets have to be imported as row data generated by the genomic sequencing chips (see Section 2.4) or the existing GWAS datasets reside in data repositories (dbGAP (see Section 2.2.3)). Before loading, the datasets need to check the access mechanism for the dataset, size of the datasets and available physical storage for the imported dataset. After satisfying the conditions mentioned above, datasets would be imported to the secure research infrastructure matched with the data-related ethical and security constraints. The importing procedures would be carried out if there was an existing GWAS data management system or the dataset was kept inside file-based disk storage.

Quality control is a major part of the GWAS data analysis because invalid data input can cause invalid conclusions in the GWAS [213]. Therefore, a number of GWAS-specific data validation procedures have been introduced and applied based on the specific research dataset, including: (1) checking the sample quality to validate the sample handling errors, (2) searching for marker genotyping efficiencies and removing the poor-quality SNPs before the analysis, and (3) searching to minimise the batch effect by maximising random genomic samples obtained for the sequencing.

Under the sample quality validation, sample handling errors are checked to exclude the impact to the conclusions by the initial sequencing process. There will be validations to check the sex inconsistencies regarding the specific chromosomal

positions. Mix up of the sex attribute in the GWAS dataset is usually validated by checking the heterozygosity rate of chromosome 23 which is used to declare the gender of the human genome. Also, there will be checks to understand the sample relatedness based on the available family information specified for the dataset because hereditary effects have a significant impact on GWAS outcomes. Population stratification plays a pivotal role in understanding the population substructures in the sample because similar population groups share the same genetic material which could be invalidly recognised as a true association allele to a trait. Similarly, in the SNP marker quality control, the effectiveness of the allele data residing in the GWAS dataset is considered. The SNPs with a high missing rate will be omitted from the analysis and their impact on the conclusions will be reduced. Control sample reproducibility is another aspect which is checked in the quality control process to eliminate rare SNPs which are removed from the analysis to avoid their impact on associations. The minor allele frequency check is used to check the rare SNPs in GWAS datasets. Finally, Hardy-Weinberg Equilibrium (HWE) is performed on the GWAS datasets to check generation effect, possible genotyping errors, and population stratification.

A quality validated dataset is essential before the analysis then selected datasets from the repository would be considered for analysis. This process would be timely and could use more of the computing resources. The SPARK chapter (see Chapter 5) discussed this process with a prospective software design. When discussing the data aspect of an analysis execution, a dataset would be a reference to the analysing package as an input and the content would be accessed based on predefined access parameters (dataset path, access specifies, data format, etc). The required constraint is keeping the GWAS analysis datasets in a secure location, efficiently managing the disk space, and enabling uninterrupted access to the dataset by the analysis program. The SPARK-specific GWAS data management design is discussed in Section 6.9.

Extracting the experiment results or the analysis output is the next important

task in GWAS data management. This includes storing the analysis result per experiment for future analysis and conclusions, enabling analysis results to be presented clearly, and sharing the experiment results with collaborators. The SPARK design has specifically provided a mechanism to store and share the experiment results (see Section 5.4). This has enabled wide secure access to GWAS analysis results by collaborating parties but required the results to be imported to third-party visualisation platforms to generate different scientific visualisations which do not require much computational power and disk space.

The above-mentioned GWAS data management process has been commonly identified as a pipeline mechanism. Researchers have developed step-by-step data processing mechanisms based on the requirement of GWAS dataset and enabled the necessary third-party tools to be supported by each step. The PLINK tool (see Section 2.4.2) has provided much of the required processing capabilities by providing data formats, quality control procedures, analysing techniques, and visualisation techniques as a standalone tool. Researchers have to import a dataset into a location which is accessed by the PLINK standalone tool and specify its functionality at a specific step by a pre-configured script. The challenging part of this approach is by developing a pipeline script addressing the total GWAS requirements and interface the process runtime exceptions, the PLINK is a single core application and enables multi-core supported high-performance computing power to execute Big Data-related processes, and provide a web-based support to monitor the analysis and access the results globally. The SPARK data management approach has provided a solution for this challenge by following novel software design-related GWAS datasets.

6.5 SPARK approach to managing GWAS datasets

To succeed, the design of SPARK must address many GWAS data management issues. To meet these challenges, the following approach was undertaken:

Benchmark the relational database table schemas to manage genomic datasets

The following approach has considered implementing normalised data schema to accommodate GWAS dataset and measure the performance parameters. Performance measurements include the insertion times of restoring GWAS datasets inside relational data schema and physical disk space obtained to hold a GWAS dataset inside relational database tables. It is an essential consideration to minimise the insertion time in a SPARK type of system to accommodate multiple GWAS datasets while simultaneously expecting a growing number of GWAS datasets belonging to multiple studies to be accommodated. Given a scenario, querying time is an important fact in extracting specific information out of existing GWAS datasets in multi-study environment. The SPARK system is designed to be used by multiple researchers who own a number of GWAS datasets. Therefore, it is important to consider how to optimise the physical disk space usage when storing multiple GWAS datasets.

Review and evaluate NoSQL database models to manage genomic data efficiently

NoSQL or the non-relational databases have been developed to overcome the limitations of relational database model (see Section 6.7). There are a number of NoSQL databases that have been developed but the most popular non-relational data models are key-value, columnar, document, and graph models. Therefore, the advantages and disadvantages of restoring GWAS dataset inside a non-relational data model will be investigated. The features of a non-relational data model will be optimised and a suitable data structure to store a GWAS dataset using the non-relational database will be proposed.

Design a software application architecture to integrate with the GWAS data model

Choosing a database model does not fully accomplish the challenge of restoring a GWAS dataset inside a database schema. There has to be an application architecture design to integrate and implement the data-related functionality required to manage the GWAS dataset. A popular data layer approach is unable to accommodate the required implementation to handle large datasets, and therefore, Lambda architecture has been used to implement required functionality to operate with large GWAS datasets. The Lambda architecture design is different to N-tier design mainly because it introduces multiple data mapping and processing tiers in application design compared to a single data tier [50]. Therefore, SPARK design incorporates a separate data mapping tier to interpret the GWAS data and processes the extraction queries as the batch processing in another set of data tiers (see Section 6.9).

The remainder of this chapter examines each of these facets in more detail. The following sections present benchmark results for both relational and non-relational database models and propose a software application architecture to integrate an appropriate database model with SPARK.

6.6 A relational approach to GWAS data management

Before implementing a new approach to genomic data management, the SPARK project tends to look into the existing genomic data management techniques. Much of the genomic datasets reside inside the file systems (e.g. PLINK text and binary) and explicitly execute analysis tools on the need for an experiment basis. Storage wise genomic datasets reside in the files which are formatted as plain text or binary versions. The PLINK formats (text and binary) are considered as defacto GWAS data formats. The PLINK text datasets consist of two files per dataset, including

MAP file to record the SNP metadata and PED file to record family and SNP data consequently to MAP file.

The PLINK binary dataset consists of three types of data files, including FAM file to represent the family information of the subjects, BIM file to present the SNP meta information similar to MAP file, and BED file to represent the binary version of the SNP alleles in the GWAS dataset. To save much of the physical disk space files are compressed (zip, tar, etc.) and decompressed on an analysis basis. This approach has been commonly used for most of the genomic dataset without considering the size. Comparatively small genomic datasets reside in the relational databases but are quite uncommon compared to the storage of large genomic datasets.

Based on the remarks mentioned above, an experiment is designed to observe the behaviour of relational and non-relational databases when storing GWAS datasets. This experiment compares the time and space taken to restore a GWAS dataset from file storage to relational and non-relational database schemas. The main advantages of restoring a GWAS dataset in a database is that it is a more organised place to keep the genomic datasets with proper storage dimensions, easy to retrieve the information when required, and more secure storage infrastructure to keep the existing genomic datasets.

When considering creating an organised data environment for the GWAS datasets, database engines have provided a predefined data schema to accommodate the datasets. This helps to organise the datasets properly into sets of tables and provide unique data schema for future datasets. Organising data into predefined tables gives the opportunity to reuse the extraction queries for multiple datasets. Also, databases have their own set of authentication and authorisation mechanisms to secure the existing datasets. Storing a dataset inside a disk only enables file system-based security provided by the operating system but stored in a database enables additional database user-specific security levels to protect the data from unauthorised access.

Therefore, the experiment considered a set of simulated genomic datasets generated by the PLINK tool. Datasets are created to cover combinations of 1, 10, 100, 1,000, 10,000, 100,000 samples/people and 1, 10, 100, 1,000, 10,000, 100,000, 250,000, 500,000 and 1,000,000 SNPs.

The simulated genomic datasets using the PLINK tool have the following characteristics. According to the previous description, PLINK tool has generated a dataset for each combination. The data generation process was carried out with n number of controls equal to the total number of subjects in the dataset and the lower allele frequency is set to 0 and upper allele frequency to 1. These parameters have been set to determine the default values to the data generation process, and these values are not involved in any aspect of the experiment results because the experiment has only focused on the present value and how these values are to be stored in the relational database scheme. Datasets have included a directory consisting of two major data files, including <FILENAME>.ped and <FILENAME>.map. <FILENAME>.map file consists of the meta information related to individual SNPs. <FILENAME>.ped file consists of the individual demographic information and relevant SNP alleles recorded in the map file. In addition to recording the SNP alleles belonging to a map file, the ped file maintains the exact order of alleles belonging to certain SNPs as its predecessor.

The total physical disk space acquired to store the given data matrix is shown in Table 6.1 below.

MySQL database has been selected as the preferred database engine to execute the experiment. MySQL database system has a long-term reputation as one of the pioneered open source databases which started around 1994. Currently, serving as the main database engine for a number of software systems developed by industry giants, it has been mentioned in numerous tech conferences, including Google, Facebook and Amazon Inc. As a database engine, MySQL supports major operational aspects, including following the ANSI SQL 99 standard, cross-platform

21	rs11511647	0	26765
X	rs3883674	0	32380
X	rs12218882	0	48172
9	rs10904045	0	48426

9	FAM1	NA06985	0	0	1	1	A	T	T	T	G	G	C	C	A	T	T	T	G	G	C	C
8	FAM1	NA06991	0	0	1	1	C	T	T	T	G	G	C	C	C	T	T	T	G	G	C	C
10	0	NA06993	0	0	1	1	C	T	T	T	G	G	C	T	C	T	T	T	G	G	C	T
8	0	NA06994	0	0	1	1	C	T	T	T	G	G	C	C	C	T	T	T	G	G	C	C
	0	NA07000	0	0	2	1	C	T	T	T	G	G	C	T	C	T	T	T	G	G	C	T
	0	NA07019	0	0	1	1	C	T	T	T	G	G	C	C	C	T	T	T	G	G	C	C

Figure 6.2: Example PLINK PED/MAP file displaying the correlation between two file sources [4].

support, database triggers, stored procedures, etc. Selection of MySQL database is supported by the Open Source license of the system, the reputation of the most popular open source database engine, and it is per file per table, implying an InnoDB data schema. InnoDB data schema would help to measure the physical disk space taken to table the dataset exactly.

MySQL database has implicitly emphasised a third normal form relational database table structure to represent the dataset. The feasibility experiment has been designed based on this data structure. Therefore, the experiment has evaluated the time taken to restore PLINK GWAS dataset in the relational table schema and actual database physical table space acquisition to store the dataset. The following diagram highlights the results of this experiment based on the execution time of the dataset and physical table space allocation for each dataset. The following entity relationship diagram indicates the third normal form table structure used to store PLINK formatted GWAS datasets.

A data restoration experiment has been executed in a dedicated computer server allocated for the analysis. The experiment execution environment consists of the following hardware infrastructure. Experiment has been executed in a Dell PowerEdge R410 server consisting of two 2.93 GHz x5570 Intel(R) Xeon(R) processors

SNP count/ Population	1	10	100	1,000	10,000
1	31	169	2,100	18,801	207,802
10	229	691	5,862	54,963	567,964
100	2,389	6,091	43,662	416,763	4,169,764
1,000	25,789	61,891	423,462	4,036,563	40,189,564
10,000	277,789	637,891	4,239,462	40,252,563	400,405,564
100,000	2,977,789	6,577,891	42,579,462	402,592,563	4,002,745,564
SNP count/ Population	100,000	250,000	500,000	1,000,000	
1	2,977,803	7,777,803	15,777,803	27,777,804	
10	6,577,965	16,777,965	33,777,965	63,777,966	
100	42,579,765	106,779,765	213,779,765	423,779,766	
1000	402,599,565	1,006,799,565	2,013,799,565	3,771,390,099	
10000	4,002,815,565	10,007,015,565	20,014,015,565	23,580,047,625	

Table 6.1: Dataset sizes according to the population and SNP count matrix generated by PLINK (bytes).

supported by 128 GB Multi-bit ECC DDR3 RAM (8 X 16 GB Simms). The physical disk space consists of four system disks supported by RAID-1 virtual disk specification. The total disk space consists of 1,488 GB storage capacity SEAGATE branded hard disks with 15,000 RPM.

The PED/MAP file processing has been executed by a Java program and written to the MySQL table schema using the Java database connection (JDBC) driver. The processing has been executed multiple times (<5 times) to determine the average time of the restoration process. Each time before execution, a database schema with a set of empty tables and processing time is re-initiated, and table data space is measured and logged automatically by a script.

Table 6.2 summarises the CPU time taken to store each dataset in a MySQL database schema. The CPU time indicates the exact time stamp taken by a CPU to run the particular processes, excluding the processor time taken by other software applications that are running simultaneously on the same machine.

Referring to Table 6.2, observe the time increase when the dataset has grown by SNP counts and population counts. The line chart displays the further trends

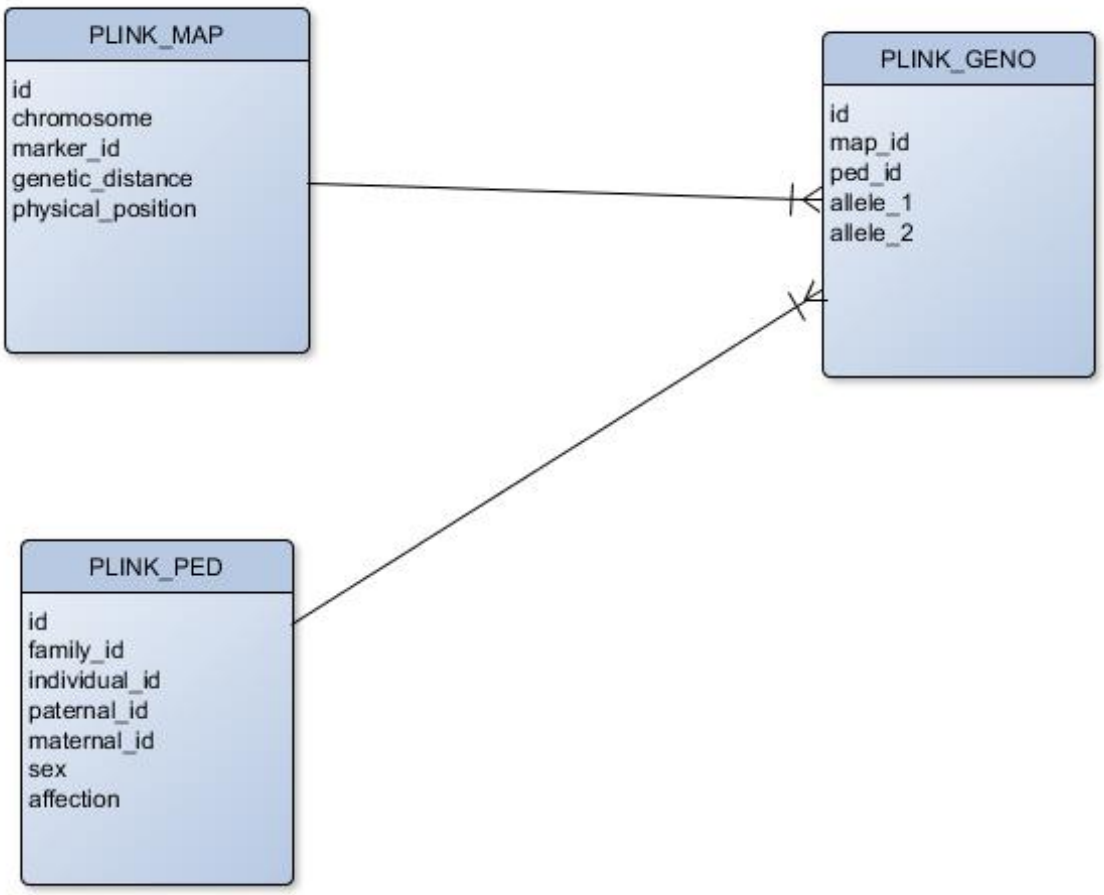


Figure 6.3: Entity relationship diagram of the MySQL database engine-based experiment table schema

against the population (1–100,000) (see Figure 6.4) and SNP counts (1–1,000,000) (see Figure 6.5).

The experiment has been carried out to examine the time taken to store a GWAS dataset in the relational data schema. Also, disk space taken to store GWAS dataset is another factor because these datasets are typically quite large, in the order of gigabytes. Based on the InnoDB schema type in the MySQL relational database, each table has its data file to record the data. Table 6.3 has summarised the total disk space allocation for the experimental datasets and disk space allocations by line charts against the population (1–100,000) (see Figure 6.6) and SNP counts

SNP count/ Population	1	10	100	1,000	10,000
1	0.5240	0.5320	0.5900	0.9860	3.3662
10	0.5310	0.5570	0.7560	2.1450	7.0725
100	0.6140	0.7970	1.9540	5.4440	43.3512
1,000	1.2620	2.6010	5.4740	32.3910	395.1086
10,000	3.7860	6.6290	34.0340	299.7560	3,888.1760
100,000	13.0360	39.6770	308.3920	2,976.1670	39,993.0300
SNP count/ Population	100,000	250,000	500,000	1,000,000	
1	12.0320	26.3720	50.6420	97.8240	
10	52.9040	132.1080	257.2020	521.7840	
100	460.7920	1,144.0960	2,364.7920	4,909.5140	
1000	4,574.7920	11,671.3520	24,065.9820	46,172.3900	

Table 6.2: CPU time (Seconds) to insert GWAS datasets into a relational data schema using MySQL database engine

(1–1,000,000) (see Figure 6.7).

The experimental observations are the increase of population count, or SNP would increase the loading time and disk space allocations. As an example, according to the table results (see Table 6.2), it is evident relational data schemas would have taken approximately 6 h to 500,000 SNP X 1,000 population dataset to store in the relational database schema. The result of more than 6 h for uploading 500,000 SNP dataset would be compelling when loading multiple datasets in a number of studies with multiple researchers in a single web application hosting server. Also, disk space allocations of the relational data schema have exceeded the file disk space allocation of the same dataset. According to the same dataset (500,000 SNP X 1,000 population) time estimation in the section relational schema disk allocation has 64,269,123,584 bytes (64269.12 MB) and file allocations have been only 2,013,799,565 bytes (2013.8 MB). In the given scenario, relational schema disk allocation is approximately 32 times higher compared to the file storage.

Considering the outcomes of the experiment, there is clear evidence to increase

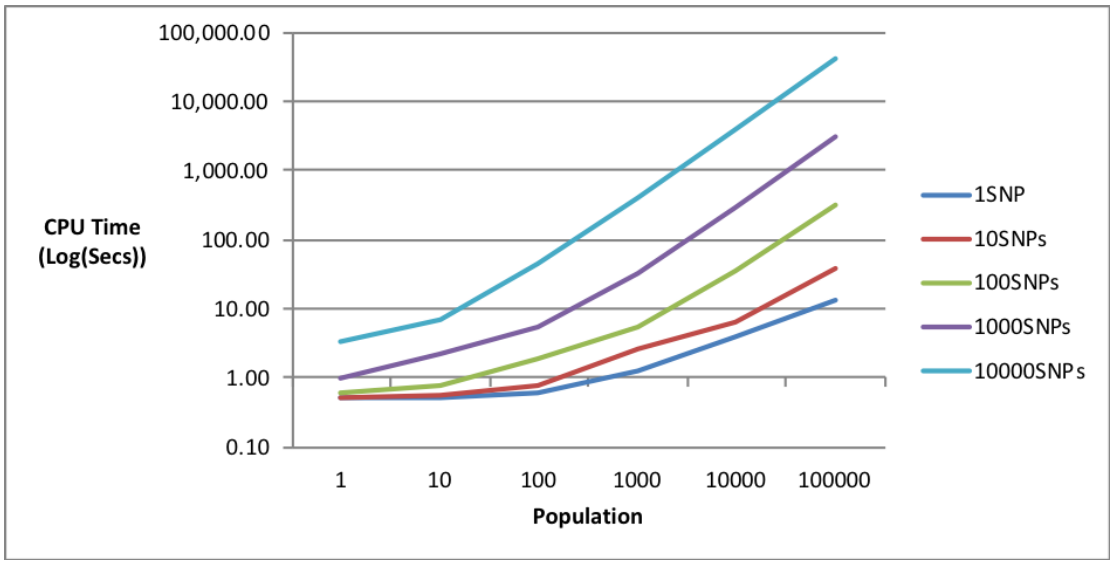


Figure 6.4: CPU time taken to insert GWAS datasets into a relational table schema using MySQL

the insert time and disk space according dataset sizes. Increasing the insert time has raised two questions. First, is the time taken to restore a GWAS dataset feasible for web application space? Web application designers will have to consider the longer wait time to complete a GWAS data uploading task (according to the previous example >6 h) which already exceeds a single web session (allowable time limit to remain on a web page without a user interaction), even though implementation as a batch uploading has to consider the large background job executions in web space and the implications on multi-user and multi-study management systems when executing the simultaneous uploading tasks. Second, is the disk space obtained to store GWAS dataset in a relational data schema worth sparing? Because disk space allocation in normalised relational schema files systems is much higher than the original GWAS file system storage and close to 32 times higher than that considered in table datasets (see Tables 6.1 and 6.3).

Results have indicated restoring the large genomics datasets in a relational data schema is challenging for designers. The time taken to restore a genomic dataset in a normalised relational table structure is impractical. Also, it is inefficient to restore

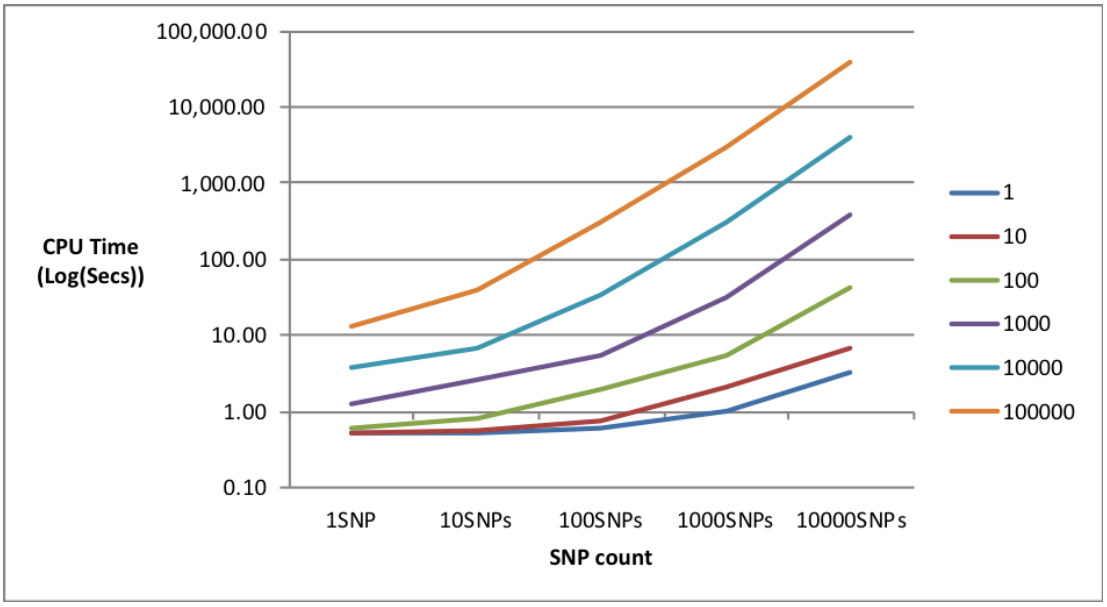


Figure 6.5: Line chart representing the CPU time taken to store SNP datasets in relational table schema against the SNP count

a GWAS dataset in a relational database schema as a large amount of disk space is required. The only feasible argument supporting the genomic data migration to a relational database is to keep the datasets in a secure and well-maintained data management system. The above factors are the key constraints of storing and managing a dataset in relational database management systems. It is a significant challenge to owners of the GWAS datasets to manage the existing datasets in a secure and efficient data management environment. This is a common problem when storing large datasets in relational databases. Another technology has been developed to address this challenge called the NoSQL databases. NoSQL databases take a substantially different approach to model data of Big Data scale, and these will be discussed in the following section.

SNP count/ Population	1	10	100	1,000	10,000
1	<1	<1	<1	131,072	9,748,480
10	<1	<1	81,920	9,338,880	22,358,752
100	<1	81,920	9,289,728	21,949,152	117,458,752
1,000	81,920	9,289,728	24,000,000	142,049,152	1,300,458,752
10,000	9,289,728	20,800,000	113,000,000	1,010,049,152	9,950,458,752
100,000	20,800,000	109,000,000	981,000,000	9,680,049,152	96,700,458,752
SNP count/ Population	100,000	250,000	500,000	1,000,000	
1	34,357,248	55,328,768	96,223,232	167,526,400	
10	135,020,544	308,035,584	595,345,408	1,169,965,056	
100	1,049,378,816	2,598,125,568	5,171,331,072	10,317,742,080	
1000	12,868,976,640	32,142,852,096	64,269,123,584	118,274,981,888	

Table 6.3: Size (bytes) to store GWAS datasets in a relational table schema using MySQL database engine InnoDB schema

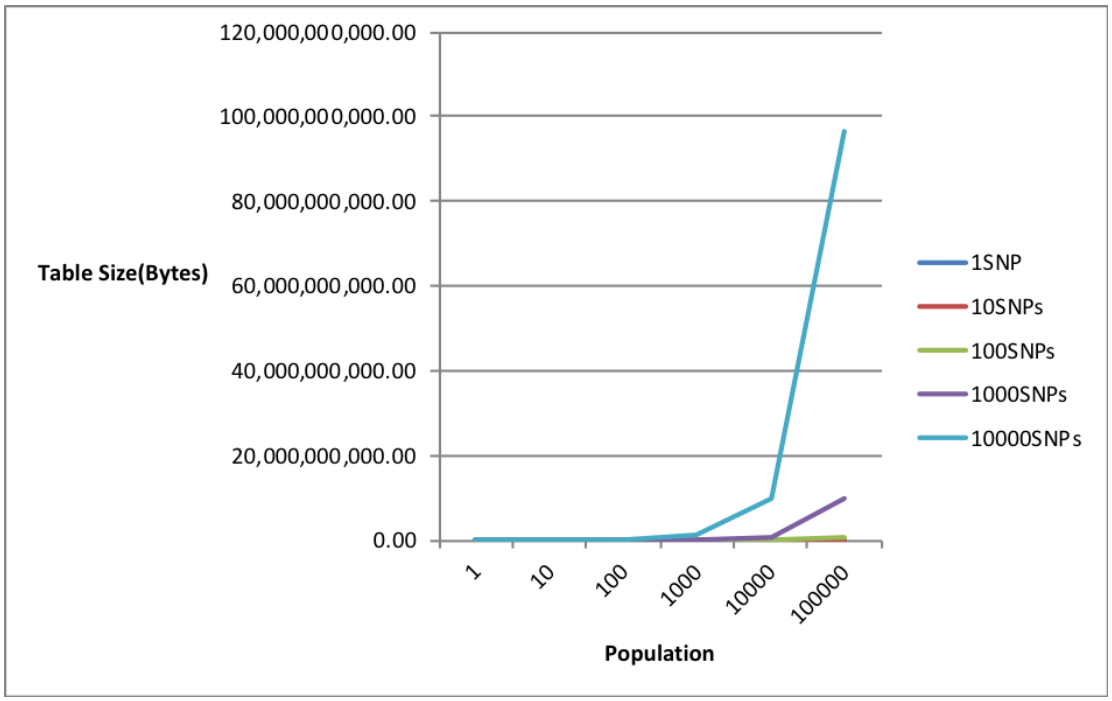


Figure 6.6: Disk space required to store datasets in a relational table schema using MySQL database engine InnoDB schema

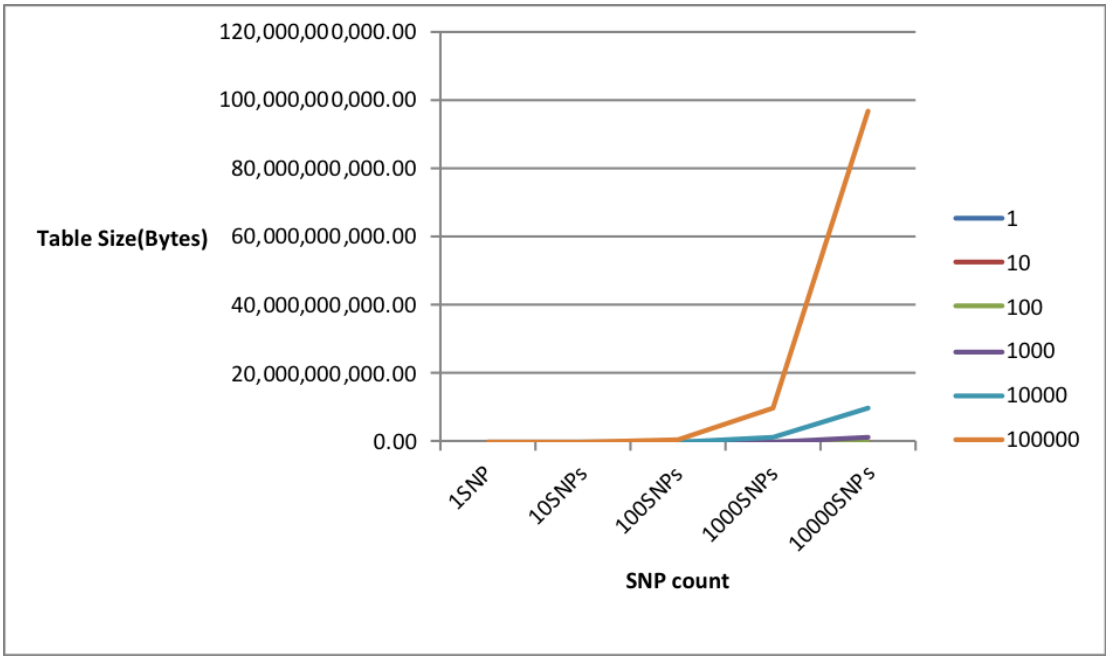


Figure 6.7: Disk space required to store GWAS datasets in a relational table schema using MySQL database engine InnoDB schema

6.7 Non-relational approaches to GWAS data management

The introduction of the non-relational databases, also known as **NoSQL** databases, to manage Big Datasets led the researcher to investigate the feasibility of storing large heterogeneous biological datasets in them. NoSQL databases follow different data models compared to relational databases. Relational databases follow a set-oriented data model to maintain the normalised relationship between the datasets in the tables. NoSQL databases follow non-relational data models which are suitable for the datasets and do not fit into the normalised context and exceed the capabilities of the relational data model. Therefore, researchers have conducted feasibility analyses in various NoSQL database models to check suitability to accommodate genomic Big Datasets, including key-value, document, columnar and graph types [214]. It would be interesting to discuss the nature of NoSQL database

models to understand their applicability to GWAS management and potential application to the goals of SPARK. The main types of NoSQL data models are now discussed.

6.7.1 Types of non-relational data models

Key-value model

The key-value model stores information based on specific id (key) and value attached to the identity. A single value attached to a specific identity is always stored. This value could be another id representing a composite primary key. In programming terms, the key-value data model exactly represents a hash map where a unique identifier is assigned to a value residing in a map. Calling the unique id would return the value assigned to it. In the case of a dataset nature data elements which can be presented in key-value pairs are the best fit scenario for a given data model. GWAS datasets are usually represented by multiple allele attributes per individual which may exceed millions of key-value pairs. Therefore, choosing a key-value model to store the GWAS datasets would make repeatable key-value pairs consecutive to the genomic attributes. Key-value-based NoSQL databases, Redis[3] and Riak[4], are available systems for the application development.

An example with a PLINK-based GWAS dataset consisting of 1,000 study participants with 100,000 allele pairs is presented. A PLINK-based GWAS dataset consists of two types of information, including SNP details (text-based PLINK MAP file) and individual SNP details (text-based PLINK PED file) for all the study participants. The SNP details for this example include a set of key-value pairs starting with a unique id for each SNP in the MAP table (see Table 6.4). MAP tables have a unique key value for the individual SNPs and each value contains the individual SNP properties. Similarly, PED tables have the individual

[3] `https://redis.io/`

[4] `http://basho.com/products/riak-kv/`

Key	Value
SNP Id1	Chromosome Id, Physical position, Genetic distance
SNP Id2	
SNP Id3	
SNP Id4	
SNP Id(n)	

Table 6.4: MAP table format

Key	Value
Individual Id1	Family Id, Paternal Id, Maternal Id, Sex, Affected status
Individual Id2	
Individual Id3	
Individual Id4	
Individual Id(n)	

Table 6.5: PED table format

identity as the key value and each value contains the individual demographic information (see Table 6.5). For the genomic data, there is an allele table representing the allele combinations (see Table 6.6). Finally, for each SNP (X) a table of key-value pairs representing the subject id as the key and value for relevant allele id of the given SNP (X) is provided (see Table 6.7).

By using a key-value-based data model, a clean, structured, and easily understandable representation of the GWAS datasets in a NoSQL database is provided. Even though the following presentation has enabled a logical data model for the GWAS datasets, it is quite difficult to handle a large dataset in a given context. According to the earlier example, this would create 100,000 key-value-based tables to represent individual SNP allele for the 1,000 participants. Increasing the number of tables for a dataset would make it difficult for the data managers to manage and would increase the extraction queries for such datasets.

Key	Value
0	aa
1	00
2	AA
3	Aa

Table 6.6: Allele table format

Key	Value
Individual Id1	Allele Id 0,1,2,3
Individual Id2	
Individual Id3	
Individual Id4	
Individual Id(n)	

Table 6.7: SNP(X) table format

Document model

Document model stores information based on a predefined document template. Naturally designed to handle the documents-based datasets (Text, Word, Excel, HTML, and XML type documents) in a database environment without preprocessing and changing the existing nature of the dataset. Each template consists of a document format which includes its attribute types and constraints. Datasets have been organised inside the containers based on JavaScript Object Notation (JSON) per file. Similar to representing a document with a specific format, the data is recorded based on a predefined format. Therefore, recording the GWAS datasets inside a document template could be very costly based on their repetitiveness and size. There are examples of storing the GWAS datasets in the document databases as binary objects rather than their actual presentation [6]. The given experiments presented by [6] have shown an advantage in insertion and extraction operations over the relational databases but are unable to provide space to manage the genomics datasets inside the efficient container.

An example of a GWAS dataset in PLINK format and their representation inside a document model is presented. There would be two documents matching with MAP and PED file properties represented in the JSON format. The MAP docu-

ment would be similar to image (see Figure 6.8) with SNP property attributes per file. Similarly, PED files will represent the individual demographic attributes with the genomic alleles (see Figure 6.8). The document model approach would provide a file-oriented pre-structured schema for the GWAS datasets in a secure database space. This helps to manage and extract the GWAS data via a programmable interface and use in an analytical process for further investigation. When considering the disadvantages, the document model is limited by the size of the single document. Each document size is limited to a database-specific parameter and large files would be stored as the distributed documents. The reason for this limitation is the original design of the document containers are based on the usual document files rather than GWAS-type large files. Therefore, storing as distributed documents would be similar to the binary object in the relational databases and remove the document-related data processing advantages. In the application development context open source-based MongoDB[5], and cloud-based Amazon Dynamo DB[6] are quite a popular document model-based databases available in the industry.

Columnar model

The columnar data model-based databases are specifically designed to manage the repeatable data sequence in a table structure. The large data sequences are common properties of a Big Data environment generated by sources like weather streams; news streams type continuously updating sources. Common characteristics of such a data source are long repeating text characters set either in alphanumeric or ASCII characters. Storing such datasets in a relational database table structure would create scaling issues such as exceeding the capacity of a single table column, unable to declare predefined table schema to accommodate such dataset, and leading to byte or character large object chunks to be piled up in relational tables.

Also, a relational database model is limited to a maximum column count per

[5] `https://www.mongodb.com`
[6] `https://aws.amazon.com/documentation/dynamodb/`

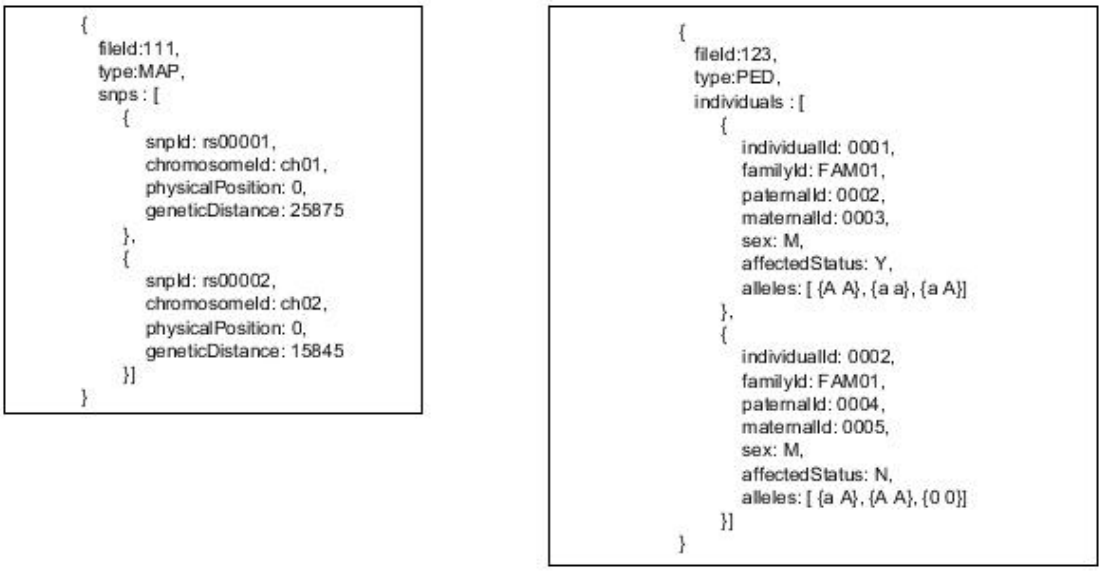


Figure 6.8: The document model representation of the PLINK-based GWAS dataset (MAP and PED file representation

table, usually, limited to a couple of thousands. For example, there is a maximum of 4,096 columns per table limit[7] for the MySQL database. To address relational database limitations to handle long character streams, columnar databases has been introduced, allowing billions of columns per table to be declared in the database schema. Allowing a large number of columns per table has sacrificed some of the qualities inherent in the relational database table. Composite primary keys, foreign key referencing to other tables, and omitting the table index creation are the notable relational table features omitted in the columnar model. Avoiding the relational features, the columnar model has enabled significant advantage over the large sequential datasets. The dataset which expands horizontally based on a single identity has the best fit for this model. In the business context, timestamp-based datasets are comparable to the columnar model. The Apache Cassandra[8] and Apache HBase[9] are much widely accepted columnar-based databases avail-

[7] `https://dev.mysql.com/doc/refman/5.7/en/column-count-limit.html`
[8] `http://cassandra.apache.org`
[9] `https://hbase.apache.org/`

able in application development space, satisfying the much demanding Big Data management issue.

Given the columnar context, sequential genomic datasets have an advantage over being storing inside a multi-column table. A genomic data record uniquely identified by the person identity belongs to its origin. The full sequence genomic record can contain up to three billion allele pairs per individual DNA sequence. In the GWAS data management space, columnar databases can play a pivotal role by accepting wider allele pairs per single table because of their default tendency to accommodate repeating character streams. Section 6.8 further investigates the columnar database approach to handle a GWAS dataset based on experiment case study ran in Cassandra database.

Graph model

A graph is a well-known data structure used in computer science and applied in multiple data modelling scenarios. Relationships among the data elements are represented in nodes and vertices. Nodes represent the distinct data elements, and vertices give the relationship between each data element. Following the graph structure, datasets are modelled based on nodes and a vertex. Individual entities mapped into the nodes and their relationships are represented by the vertex. Datasets containing the complex relationships between the data elements have an inherent advantage in the graph model. Relationships between the individual data elements are implicitly recorded in the creational time and avoid examining the relationships at the data retrieval time. Compared to the relational model, graph databases are quite efficient in retrieving the information based on the complex queries because query logic is already calculated and recorded in the graph database. The Neo4j[10] graph-based database is a popular open source solution which has been used for multiple Big Data solutions.

[10] `https://neo4j.com/`

Taking the similar example used in examining the previous data models (PLINK-based 1,000 participants and 100,000 SNP dataset) restoring the graph database can be explained. Each node records the individual SNP meta information per SNP in the MAP file and an individual participant's demographic records in the PED file. Allele nodes created per allele pair will be located in the PED file against each individual. The relationship record has been established to specify which allele represents which SNP for each individual (see Figure 6.9). Therefore, (1,000 X 100,000) allele nodes, 1,000 participant nodes, and (100,000) SNP nodes were created for the dataset and (1,000 X 100,000 X 2) relationships were created per dataset.

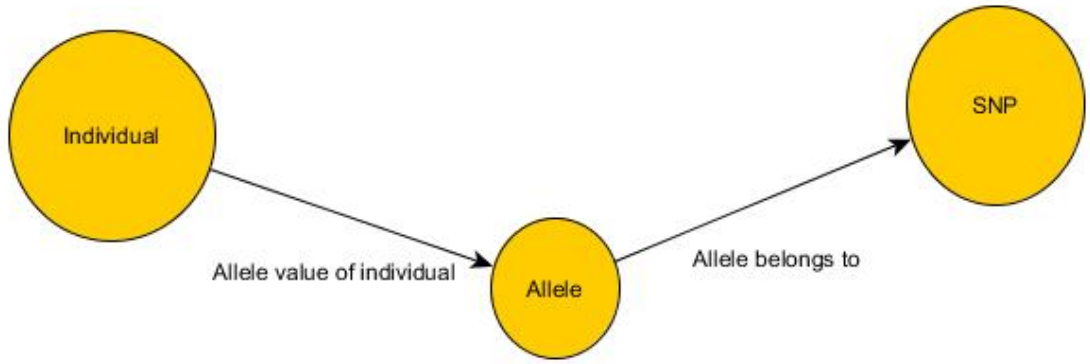


Figure 6.9: Graph data model representation of PLINK (PED/MAP)-based GWAS dataset

Representing the genomic data inside graph database makes quite an expensive data model due to the immense number of nodes (>100,000) and relationships between the nodes. Genomic datasets have repeating data elements based on the person rather than predefined relationships between these data elements. Therefore, an extensive number of nodes per GWAS dataset would have created a messy data view rather than presenting data in much cleaner and systematic way.

6.7.2 Performance of NoSQL databases when applied to genomic data

Previously, there have been experiments conducted by researchers to check the suitability of NoSQL databases to store the genomic datasets. According to the literature, experiments have been conducted in document, columnar and graph models to check their suitability for genomic data management. Individual experiments have been conducted based on data modelling technique, insertion times, and extraction times according to the query parameters. Especially in the GWAS context, multiple datasets need to be stored in the database environment; multiple search queries have to be executed simultaneously against a number of datasets, and the information extracted after completing the search criteria.

6.7.3 Document model-based NoSQL database for genomic data management

[6] An experiment based on data provenance mechanisms for the genomic analysis was conducted and the relational and non-relational database performance on storing genomic datasets was benchmarked. The experiment ran on Linux operating system with HP 8-Core 2.13 GZ Zeon processor with 32 GB RAM. Experiment creators have chosen the MongoDB document model-based NoSQL database system and PostgreSQL relational database management systems to compare the performance. A FASTQ [215] formatted cell dataset was selected, including different sizes represented in Table 6.8. A FASTQ data file includes the character stream representing the DNA sequence data associated with their quality scores. An example of a FASTQ file template is given below (see Figure 6.10).

The following files have been moved to the MongoDB as the JSON-based document streams are supported by GRIDFS storage schema. Here, files were automatically portioned into 16 MB individual document segments and stored in separate

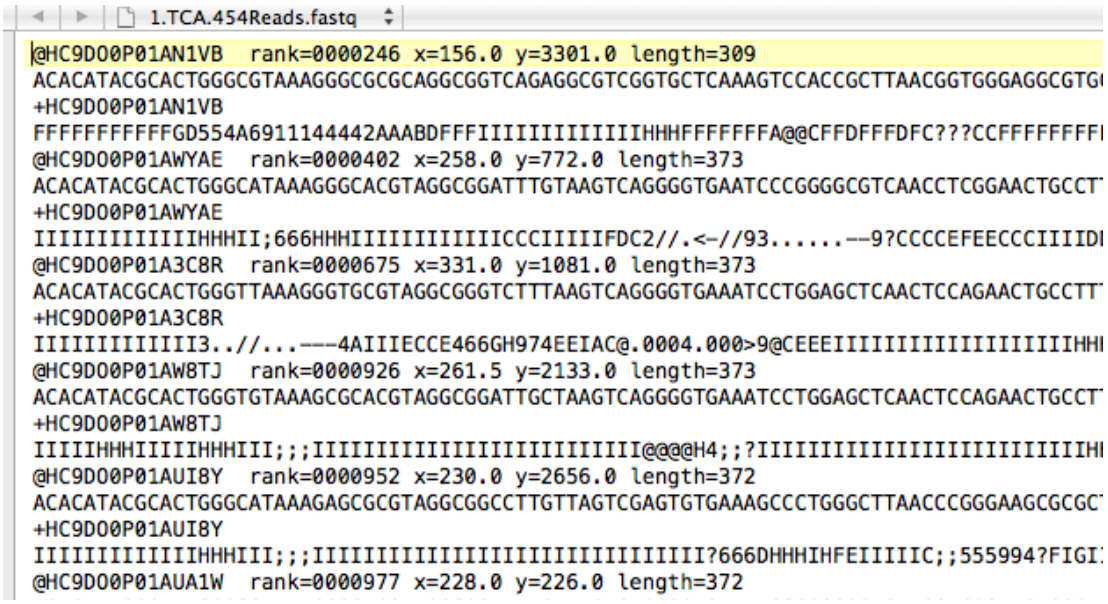


Figure 6.10: Example FASTQ file available to understand the data format. [5]

	File	Size
Liver Cell Files	Sample – 1	11.4 GB
	Sample – 2	4 GB
	Sample – 3	3.2 GB
Kidney Cell Files	Sample – 1	8.1 GB
	Sample – 2	3.8 GB
	Sample – 3	5.9 GB

Table 6.8: Sample dataset sizes used in the experiments of [6]

individual file chunks. As a comparison experiment, the same dataset was moved to the PostgreSQL database as 2 GB Byte Large Objects (BLOBs). Both representations needed processing after extraction to enable anything meaningful to be achieved with the data, especially, related to the GWAS context involved in querying, cleaning, and filtering operations. The insertion and extraction times for each dataset in these two database engines are recorded as follows.

When comparing the insertion and extraction time for the FASTQ-based sequence data files in the relational and non-relational database environment, a sig-

Database	Insertion (hh:mm:ss)	Extraction (hh:mm:ss)
PostgreSQL (relational database)	01:51:54	00:28:27
MongoDB (non-relational database)	00:08:53	00:05:44

Table 6.9: Average insertion and extraction times (hh:mm:ss) for PostgreSQL and MongoDB databases for Table 6.8 dataset

nificant time advantage in the non-relational database systems can be observed. Comparing the insertion times (01:51:54 > 00:08:53) has clearly shown that the non-relational database examines the results 12.5 times faster than the relational database (01:51:54/ 00:08:53). In the extraction results (00:28:27/ 00:05:44) the non-relational database approach is nearly five times faster than the relational database. Comparing these results with the previous experiment (see Section 6.6), the non-relation database performance is much clearer.

Observing the experiment results, it is quite efficient to store a genomic dataset or any other large dataset in a non-relational database system. Given the facts, there is one question which arises from this experiment. Here, data has been stored as an exact match to their presence in the data files. Database systems have been introduced to overcome this practice with a more efficient data model approach, supported with query mechanisms to extract specific information based on a prestructured data storage model. Just storing the datasets as binary objects or extracting the whole document to process information has diminished these good practices which are building blocks for database management systems. Considering the scenario related to the GWAS dataset, the document model does not efficiently capture the structure of a GWAS dataset in a way that is useful or necessary to perform typical GWAS data processing actions. Post-extraction procedures to perform the querying operations related to GWAS operations need to be implemented. It is better to restore the existing dataset inside a database environment by providing a well-structured and secure meta infrastructure but the burden of post-extraction activities specified in the SPARK design (data querying, extraction, importing) needs to be avoided or minimised.

Database	Insertion (hh:mm:ss)	Extraction (hh:mm:ss)
Cassandra (2 clusters)	00:52:05	01:16:25
Cassandra (4 clusters)	00:42:56	00:53:49

Table 6.10: Cassandra database average insertion and extraction times for Table 6.8 dataset

Therefore, further examination of other non-relational data modules are important to draw a big picture of efficient genomic data storage. Columnar databases can store large data sequences as text character arrays of their multi-column approach which massively exceeds the relational database schema tables.

6.7.4 Columnar model-based NoSQL database for genomic data management

The same dataset and hardware [6] used to evaluate the document-based non-relational database have been used to evaluate the performance of Cassandra non-relational columnar database. A preprocessing mechanism has been developed to break down the large FASTQ files into a small set of files which are ready to insert into the Cassandra database tables. The given files contain 10,000 rows with 10 columns per each row. A row consists of 10 columns with five sequence values for each field. Ultimately, one sequence consists of 36 base pairs and a single column field consists of 180 bases. Every file mapped to a table resides in the Cassandra database and imports file data to the database table. According to the experimental set-up for the Cassandra database evaluation [7], there are two execution environments prepared. Cassandra is a distributed database and can be hosted in multiple clusters. An experiment was conducted with two- and four-cluster environments to check the database performance. The insertion and extraction operation performance are represented in Table 6.10 for two and four database clusters.

Database	Insertion (hh:mm:ss)	Extraction (hh:mm:ss)
PostgreSQL (relational database)	01:51:54	00:28:27
MongoDB (non-relational database)	00:08:53	00:05:44
Cassandra (2 clusters)	00:52:05	01:16:25
Cassandra (4 clusters)	00:42:56	00:53:49

Table 6.11: Summary of insertion and extraction times from the experiments of [6, 7] for Table 6.8 dataset.

Analysing the results closely (see Table 6.10), it is evident that the insertion and extraction times in the Cassandra database environment are parallel with the number of computation clusters allocated to operate the database engine. Increasing the computational clusters has reduced the insertion and extraction operation times. Another fact is that Cassandra columnar database data extraction time has exceeded the insertion time of the dataset.

To make a comparison between relational, non-relational document, and columnar databases the following table (see Table 6.11) merged the results obtained from the document and columnar database experiments.

Based on the published database comparisons (relational vs. non-relational), it is evident non-relational databases have performed adequately in managing the genomic datasets. The relational versus non-relational comparison was conducted based on a much popular relational database of PostgreSQL and two popular non-relational databases of MongoDB (document model) and Cassandra (columnar model). Comparing the insertion and extraction time of non-relational databases has significantly overridden the relational domain. In particular, the document-based MongoDB data has overridden the overhead of insertion and extraction times taken for the genomic dataset. Cassandra database has shown low insertion extraction performance against both PostgreSQL and MongoDB databases. However, the poor performance of the Cassandra database can be justified due to the specified table column-based genomic data representation rather than storing BLOBs in

both PostgreSQL and MongoDB systems. Also, Cassandra database has shown performance enhancement parallel to adding more database clusters and extracting the processing power of additional computing nodes. Therefore, non-relational databases are much better suited for the genomic data management. Selection of a suitable non-relational database for GWAS data management needs to be evaluated further based on following facts observed from the previous data experiments. SPARK design is focused on providing an efficient and real-time solution for managing the GWAS dataset by enabling querying, filtering, and export options.

6.8 An efficient GWAS data encoding for the columnar data model

Previous experiments have indicated NoSQL databases outperform the existing relational databases due to their insertion and extraction operations (see Section 6.7). Efficient time management has raised the question of physical disk space obtained by the GWAS dataset in a non-relational database environment and non-relational database capabilities to efficiently store the GWAS datasets. Suitability of selecting non-relational database model to manage the genomic datasets depends on the following facts: (1) the amount of time taken to insert the existing genomic datasets into the non-relational database, (2) the amount of physical disk space is required to store a genomic dataset, and (3) specifically, the suitability to manage the GWAS-based genomic datasets. An experiment was developed to compare the Cassandra NoSQL database to a relational database to manage existing GWAS datasets because it has a distributed column approach to managing the character sequences.

According to the previous explanations in Section 6.7, MongoDB document database and Cassandra columnar database have been chosen as the most efficient options. MongoDB is able to store GWAS datasets as it uses the binary objects in its self-segmented storage mechanism. The MongoDB approach is similar to

storing large files as BLOBs in the relational databases. Insert operations are more economically viable in document storage models followed by MongoDB (see Section 6.7). However, the relational and non-relational approach is still quite efficient compared to keeping files in the disk file system as an archive, or in the original format because a database provides a more secure data storage infrastructure and inherent data organising capabilities to the files (see Section 6.4). Keeping the files as large BLOBs will restrict the total capabilities of database management system which are utilised for GWAS management. Keeping data in a binary storage is efficient for managing the data types as images or PDF documents. GWAS dataset files have character sequences along with repetitive behaviour containing millions of character sets. Therefore, the columnar data model has been chosen to check the applicability of efficient data extraction via query operations inside the database engine.

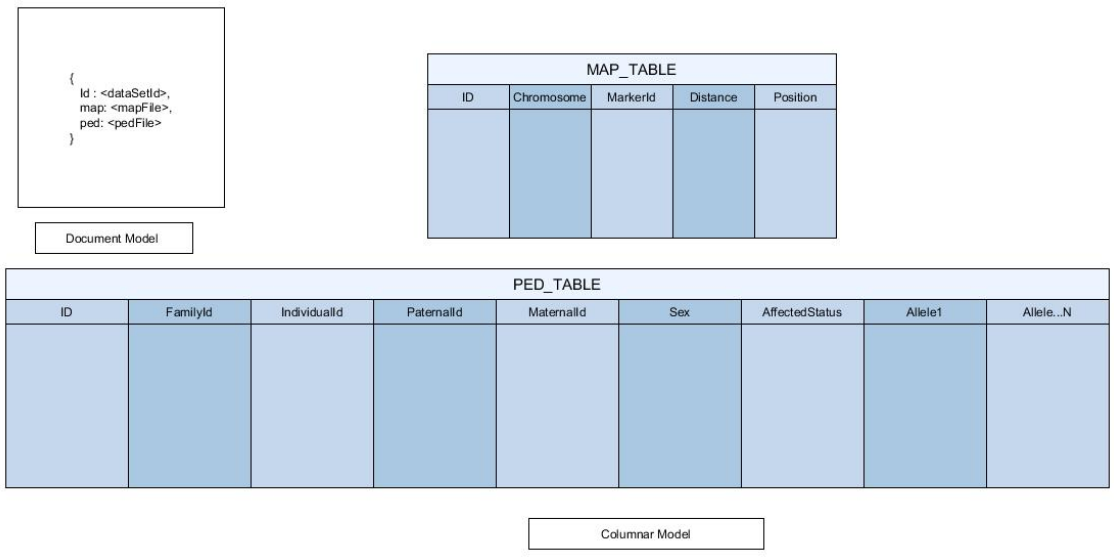


Figure 6.11: The document and columnar non-relational data containers to store PLINK-based GWAS datasets and their logical representation

To maximise the advantage in saving the physical disk space used to store a GWAS dataset and enabling the data query capabilities in the database engine, Cassandra columnar database system was chosen to enable an efficient data storage

infrastructure. The same dataset used to analyse the capabilities of relational database systems was chosen (see Table 6.1).

Figure 6.12 table schema has presented the experiment table schema designed to store PED/MAP-based GWAS dataset in the Cassandra table schema. MAP file has been transformed into `MAP_TABLE` exactly matching the tab separated content (CHROMOSOME, MARKER ID, GENETIC DISTANCE, and PHYSICAL POSITION) along with a row identity as a primary key. The phenotypic information contained in the PED file has been separated into a single table named by `PED_TABLE`. The pheno table includes the following columns identified by a primary key (FAMILY ID, INDIVIDUAL ID, PATERNAL ID, MATERNAL ID, SEX, and AFFECTED STATUS). In the diagram, allele information belonging to individual subjects resides in the `GENO_TABLE` representing per allele pair per column.

The genomic information is typically the most sizable part of a GWAS dataset. As a relational model-based table structure, it is possible to record the allele pair per column or each allele per column. GWAS datasets can exceed millions of allele pairs, and columnar databases can accommodate two billion columns per table. The following diagram will explain this scenario (see Figure 6.12).

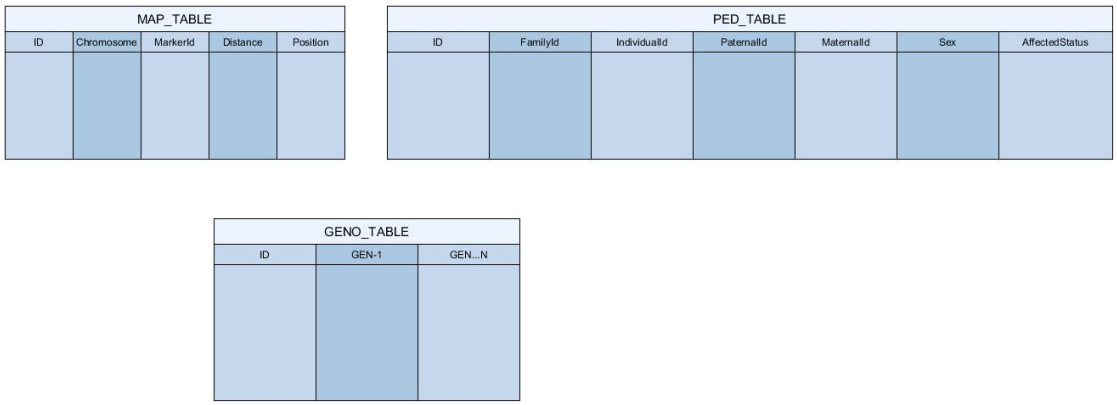


Figure 6.12: According to the multi-column model storing the GWAS MAP/PED-based genotypic data in columnar database tables

Representing the PED file genomic information has triggered the question whether

Genotype	Binary Encoding
Homozygote (AA)	00
Heterozygote (Aa)	01
Homozygote (aa)	10
Missing Genotype	11

Table 6.12: PLINK genotype mapping with their binary encodings.

this is the most efficient way of storing the genomic information in the columnar data model. In a PED file allele information has been represented by characters and each allele pair consist of two characters. Each character takes 8 bits in a standard encoding of the character alphabet, when in fact there are only four symbols that require encoding. The four symbols can, therefore, be encoded with 2 bits each, and these pairs of bits can group into lots of four so that 1 byte (8 bits) listlessly encoded four allele pairs. As explained in Section 6.8, the columnar module has an advantage in creating multiple columns exceeding the limits of a relational table and enabling feasible time frame to insert and extract the GWAS datasets.

PLINK analysing package has introduced more sensible, efficient encoding of the allele data, and this mechanism is called the PLINK binary format. The PLINK package has transformed the genomic data contained in the PED file into 8-bit formatted binary segments based on a predefined genotype group map. The following table presents the allele representation based on four different genotypes with their binary encoding values (see Table 6.12). In addition, a byte is convenient for input/output, which is easily programmed regarding bytes, but more difficult and inefficient to implement at the individual bit-level. It is also convenient as a character is a primitive data type in many programming languages, e.g. C and C++, and in those languages, it is 1 byte of information. Many languages also offer a primitive byte type.

Following the genotype encoding with binary values, PLINK creates 8-bit binary numbers with four genotypes comprised of the single byte value. The given example

has visualised the PLINK bit encoding mechanism based on the existing genotype record and this has been reiterated from PLINK documentation[11]. A binary file is simply a sequence of bits (0s and 1s) created by some software. PLINK accepts binary files as input and can produce outputs in the same format. The format of the binary file must be known to both the software producing the file and the software accepting the file as input.

According to the PLINK documentation, following PED/MAP-based GWAS data sample has been considered and converted to a compressed dataset with FAM/BIM/BED file combinations.

1	1	0	0	1	0	G	G	2	2	C	C
1	2	0	0	1	0	A	A	0	0	A	C
1	3	1	2	1	2	0	0	1	2	A	C
2	1	0	0	1	0	A	A	2	2	0	0
2	2	0	0	1	2	A	A	2	2	0	0
2	3	1	2	1	2	A	A	2	2	A	A

Figure 6.13: Sample PED file

1	Snp1	0	1
1	Snp2	0	2
1	Snp3	0	3

Figure 6.14: Sample MAP file

The dataset presented in plain text files is transformed to a combination of text and binary files to archive the dataset. The FAM file contains the phenotypic information of first six columns of the PED file.

The BIM file has extended the MAP file and has specified the allele names.

The rest of the genomic information resides in the PED file has been transformed to 8-bit binary numbers based on BIM file allele encodings and their representation in the PED file.

[11] `https://www.cog-genomics.org/PLINK2/formats`

1	1	0	0	1	0
1	2	0	0	1	0
1	3	1	2	1	2
2	1	0	0	1	0
2	2	0	0	1	2
2	3	1	2	1	2

Figure 6.15: FAM file representation of the dataset derived from the sample PED file

1	Snp1	0	1	G	A
1	Snp2	0	2	1	2
1	Snp3	0	3	A	C

Figure 6.16: BIM file representation of the dataset derived from the sample PED/MAP files

01101100	00011011	00000001	11011100	00001111	11100111
	00001111	01101011	00000001		

Figure 6.17: The BED file representation of the allele information derived from the sample PED/MAP files

Based on the PLINK binary compression, genotypes in a genomic dataset which reside in the PLINK-based GWAS dataset have been used to check the physical disk storage acquisition in the Cassandra database. Considering the Cassandra database supports data types, it does not offer a byte column type. Similar to 8-bit compression of the PLINK package, Cassandra table columns represent genotypes in 32-bit formatted integer values. Following the same allele map table (see Table 6.12) referred by the PLINK package, Java client translates the allele information residing in the PED files and makes 32-bit binary chunks compromised of 16 allele pairs per column. Before insertion of the dataset has begun, the number of columns per table are programmatically calculated to present the dataset by dividing the number of allele pairs per person by 16. Data insertion comprise integer values which consist of 16 allele pairs represented by binary encodings. In the PLINK binary files, Cassandra data schema consists of `FAM_TABLE`, `BIM_TABLE`,

SNP count/ Population	1	10	100	1,000	10,000
1	14,167.80	15,254.60	25,669.00	130,693.00	14,167.80
10	14,902.00	16,015.80	26,673.00	132,764.00	14,902.00
100	22,005.60	23,800.20	40,314.40	205,732.00	22,005.60
1,000	94,520.60	101,699.00	170,669.00	880,092.20	94,520.60
10,000	868,174.80	923,170.00	1,484,542.80	7,040,164.60	868,174.80
100,000	8,864,313.20	9,344,733.60	14,443,634.20	64,578,286.80	8,864,313.20

Table 6.13: Disk space required for GWAS dataset storage using a Cassandra database (bytes)

and `BED_TABLE` tables to restore a PLINK-based GWAS dataset. Figure 6.18 has summarised this process by visualising the genotypes of a FAM/BIM/BED files and their representation in the Cassandra database.

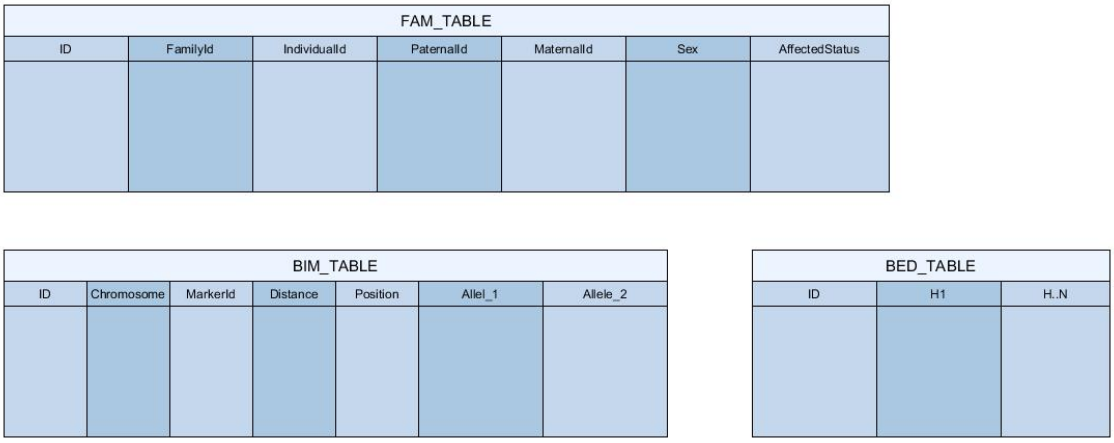


Figure 6.18: PLINK binary file representation using the Cassandra columnar model

The experiment has run for the dataset which ran the same experiment for the relational database (see Table 6.1) and results are shown below (see Table 6.13).

When comparing the earlier results (see Table 6.3) with the MySQL relational database, there has been a significant saving in physical disk space inside columnar-based Cassandra representing the datasets. A much closer look at the results has

	Relational	Non-relational
1	128.00	127.63
10	9,120.00	129.65
100	21,435.00	200.91
1,000	138,720.00	859.47
10,000	986,376.00	6,875.16
100,000	9,453,173.00	63,064.73

Table 6.14: Comparison of disk space obtained 1,000 SNP sample in MySQL relational and Cassandra non-relational databases (bytes).

demonstrated the capacity of the non-relational database in storing large datasets with their scalability. An example comparison has been carried out with the 1,000 SNP GWAS data sample represented in the relational and non-relational database environment.

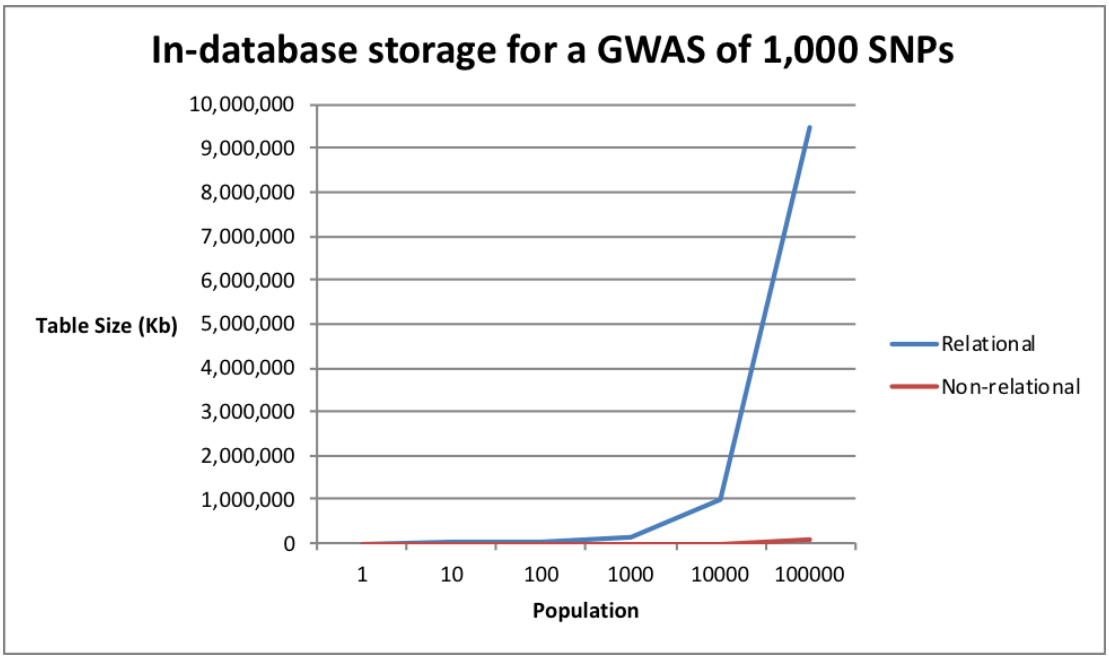


Figure 6.19: Line chart representation of disk space obtained 1,000 SNP sample in MySQL relational and Cassandra non-relational databases

The following examples have the advantage of using the Cassandra database with physical disk space saving.

The experiment results have shown the significance of storing the GWAS datasets using non-relational databases which greatly improved insertion times compared to the relational databases and provided more efficient physical disk space saving with compressed data formats inserted. The most significant implication of this experiment is the extensive use of data processing and improving the performance of data handling. The disk space saving is significant due to the processing of GWAS data into an application-specific allele encoding.

Therefore, the traditional application development with one-to-one table data mapping with the application layers have an alternate design to SPARK-specific design for GWAS data management. There is no direct data mapping in SPARK design, which consists of multiple intermediary processes in inserting and extracting the GWAS datasets from the columnar database, including encoding allele information into integer-based number chunks, choosing specific allele information based on its representation in a particular number column, and decoding the extraction results via SPARK-specific logic.

6.9 A data modelling architecture for GWAS data processing

Standalone software applications were written to run the experiments presented earlier in this chapter. However, there is a significant challenge to integrate these technologies web-based software systems, such as SPARK. This section will examine some of the architectural approaches that could be used to integrate GWAS data management with SPARK.

N-tiered software architecture is a popular design approach followed by the multiple implementation layers dedicated to each task (see Section 5.3.2). Each application tier is responsible for a specific programming task set. As in example 3

tier architecture, there is one tier for application input/output task-related implementation, another layer for the business logic implementation, and a data tier to map with database operations and manage database connections. The given N-tier approach organises the application source code into specific sections specialised in application tasks. The individual SPARK nodes have followed the N-tired software architecture for their implementation. In the N-tired software design, a data tier is responsible for the database transaction management, mapping the data table properties to application properties, and querying the data from a database based on the previous application data mapping. The given facts are perfectly valid for the relational databases which deal with data sources with foreseeable size and complexity. Big Data-oriented, non-relational databases have diminished the facts of transaction handling, application mapping and single query to extract the datasets. The given Big Data exceeded the memory capacity of application runtime to hold a record or carry out the query operations due to the large size of the dataset.

Separate data layers are diminished when dealing with the Big Data and therefore Lambda architecture has been introduced to manage the Big Data operations in a software application [50]. Separate layers per each aspect of the Big Data management have been introduced, including a batch processing layer per large data operations, a data service layer for complex data operations, and a query layer to build the data extraction functionality. In simple terms, the Lambda architecture has expanded the data layer operations into multiple application tiers specialised in individual data operations (see Figure 6.20), including a separate set of data-specific processes (Process 1–4) to carry out the data-related operations when extracting a set of information from the database. As discussed under the N-tier architecture data tier, this would be a single process to map the database table properties with application implementation and carry out the data-related operations (Create, Retrieve, Update, and Delete (CRUD) operations to database data).

The Lambda architecture has provided a loose coupling between the database

properties and application implementation. One-to-one mapping of the database properties with the application implementation has been avoided by introducing the information wrappers and avoiding the burden of the transaction handling. In particular, Big Data-related metadata to be mapped with the application implementation is overhead in memory management due to their vast sizes. Also, database transaction handling is impractical with the Big Datasets due to the complex data structures and size of the dataset. The Lambda architecture provides a solution to this challenge by introducing multiple data-related processes specialised in data-related operations without interfering with the total application runtime. As an example, there will be background processes to extract the information from the database, encoding and decoding the datasets, and querying the extracted information supported by third-party applications.

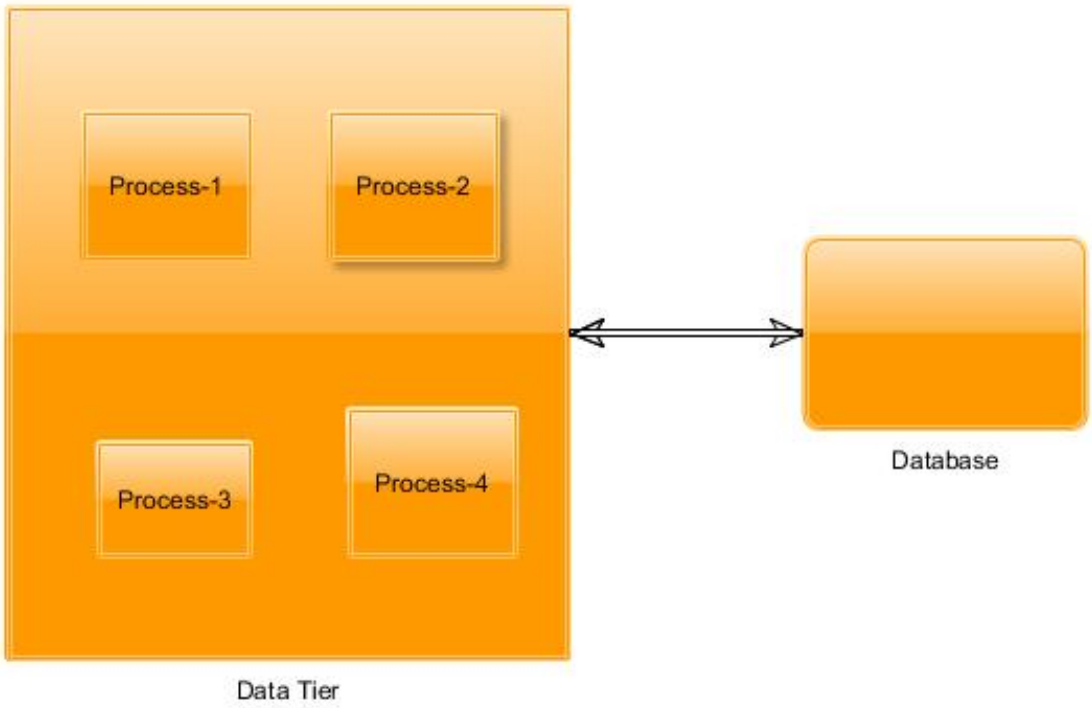


Figure 6.20: A Lambda architecture consists of multiple data processes compared with single data process in data-tier architecture

The concept of Lambda architecture best practices has been merged with the

SPARK design by (1) defining a job tier to accept data operation requests, (2) a service tier to amalgamate the data services, and (3) a data operation tier to encode and decode the data values and build the query operations based on data requests (see Figure 6.21).

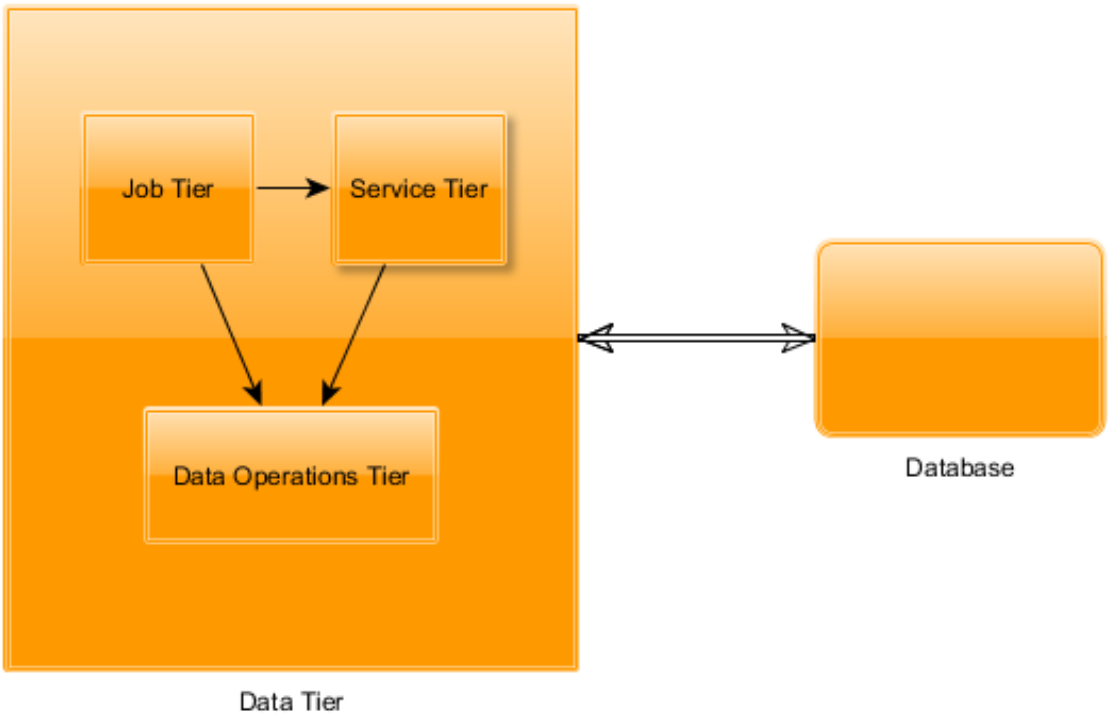


Figure 6.21: The SPARK data architecture based on the Lambda architecture guideline with multiple data tiers specialised for individual data operations

The given workaround has demonstrated the capability to manage the large GWAS datasets in a web-based software application more efficiently.

Define the job tier to accept data operations requests

Each data operation has been mapped to a job queue. Data operation mapped to a job queue event is quite uncommon in typical data-tier implementation. Typical data-tier implementation has mapped the data operation to be conducted in real-time interaction with the database and response with the data operation out-

come. Working with large GWAS datasets or the Big Data does not provide the opportunity to streamline the data operations for a request because they take a significant amount of time to complete. The time taken to complete large data operations mostly exceeds the time allocated for a typical web operation and hangs the system for unacceptable time period. To avoid unnecessary delays in data operations, there has been a job tier introduced in the Lambda architecture which enables asynchronous data processing. There will be an immediate handover of the application control to the user after requesting an operation to be executed rather waiting for it to be completed. The requested application process will be run as a background process and completed without request notification.

Service tier to amalgamate the data services

Typical software implementation has implemented CRUD operations over the data entities. CRUD operations are feasible for a dataset follows the relational data model. There are operations comparable with the data elements based on Structured Query Language (SQL) and operations in line with data elements in the database. The non-relational data model does not facilitate such flexibility with the data operations and operations bound to the programming interface provided by the database engine. Therefore, a separate data service layer was added to the data-related complex operations. When considering the GWAS data management, the most important data operations are the create, retrieve, and delete operations. Most of the time GWAS datasets reside as file structures, require one-time import to the database management system, and are deleted at the end of the analysis. However, there are multiple retrieve operations related to the imported dataset throughout the GWAS operational lifetime. The service layer is ultimately responsible to accommodate the program CRUD code that interfaces Cassandra.

Data operation tier to encode and decode the data values and build the query operations based on data requests

Data elements in the GWAS dataset have been represented in the binary encoded format in the Cassandra tables. Therefore, the mapping tier has been implemented to encode and decode the data elements in the processing. There is a separate data dictionary maintained to map the meta elements in the data tables. A common query engine was implemented to generate the query parameters and query templates per data request. The importance of having a data layer in the SPARK design is that every time a data-related request is received it has to be filtered through the data operation tier. Either request to create a new dataset in the Cassandra database has to be streamed through the data operation tier to convert the allele information to the SPARK-specific allele encoding format. Similarly, retrieval operations carried out based on predefined queries have to be transformed to the SPARK-specific allele encoding and the results translated to the PLINK-based output types.

Based on these three objectives, the following example scenario would explain the workaround based on Lambda architecture. A GWAS dataset stored in the data centre was available in the declared micro service (see Section 5.4.2). By the request of the researcher, a SPARK node receives a request to enable the dataset for analysis. First, a batch process to import the dataset to the database asynchronously will be initiated. According to the communication type defined (SSH, FTP) for the data centre, the import process will start streaming the data source to the SPARK container defined in the service tier. There, the stream will be processed and imported to the database according to the SPARK allele encoding map. When the import process has completed, a record of the particular data source ready for analysis will be made. When a query request has been received, another background process similar to import scenario is initiated then required service calls are made with the allele encoding map to deliver PLINK-based result output. This scenario explains a typical Lambda-based data operation which has

been designed and implemented in the SPARK system. Due to the Lambda-based software architecture design, a more robust and available GWAS data management via a web user interface is enabled (see Section 6.9).

6.10 Conclusion

This chapter has discussed the characteristics of genomic datasets and explained how GWAS datasets differ to many other types of epidemiological research data. Section 6.2 highlighted that the sheer size and high-dimensionality of modern GWAS datasets require specialised data management approaches. Following a detailed description of GWAS dataset properties there was a discussion of the steps involved in GWAS data management process in Section 6.4. In Section 6.6, that traditional relational database systems are impractical for GWAS data management is demonstrated; the time required for fundamental operations such as dataset insertion grows into the order of hours and days for typical GWAS datasets.

To address these challenges, Section 6.7 examined a number of non-relational data models and their applicability to GWAS data. The columnar data model was shown to perform well; both regarding speed and dataset size-on-disk, and therefore the Cassandra columnar database is selected for integration with SPARK. A more efficient data encoding for allele data within columnar columns was proposed in Section 6.8. This is particularly important as allele data is the largest component of a GWAS dataset, typically by orders of magnitude compared to the other components (phenotypes, SNP identifiers, etc.). The proposed encoding improves upon the naïve character-based representation compared to the increase of dataset sizes (see Figure 6.19).

Finally, Section 6.9 examined software architecture approaches to implement a data management module that is integrated with SPARK. The cleanly separated tiers of functionality of the Lambda architecture were found to be a good fit with SPARK's data management requirements and the columnar database model. A

Lambda design comprising three tiers was proposed: (1) a job tier to handle the data operation requests, (2) a service tier to amalgamate the data services, and (3) a data operation tier to encode and decode the data values and built the query operations based on data requests.

Based on the benchmarking results of relational and non-relational database schemas, a columnar type NoSQL database backend is selected for storage, supported by Lambda application architecture. Based on the non-relational data model and Lambda architecture, SPARK provides efficient data management to GWAS datasets (see Section 6.9).

Evaluating the SPARK design and implementation based on the usability prospective of the biomedical research domain compared to popular GWAS analysis platforms.

7 SPARK Evaluation

7.1 Introduction

This chapter serves to introduce the evaluation of SPARK design compared with enterprise systems and existing GWAS analytical platforms. The different scenarios of introducing a software platform for an operational environment will be discussed. A wide variety of literature is available for introducing an enterprise system for the operational environment rather than the research domain. The Ark is a biomedical data management platform which strongly correlates to existing enterprise system by its functionality, and SPARK design is based on the existing study-based data management system. The common attributes identified for an enterprise system were evaluated and benchmarked against popular GWAS analysing platforms called

SeqWare and Galaxy which were discussed earlier in the literature review chapter.

The SPARK-specific evaluation section discusses the in-depth analysis of design approach against the SeqWare and Galaxy platforms based on enterprise system evaluation criteria. Seven evaluation attributes identified by the enterprise system evaluation criteria were discussed. The functionality evaluation compared the SPARK-based GWAS study management with the SeqWare and Galaxy systems. Similarly, the reliability and cost of choosing the SPARK system for a default GWAS management platform was benchmarked. The ease of customisation and adaption to the operational environment, the reputation of the software vendor, and how easy to implement the SPARK-based GWAS data management process compared to existing systems is discussed. Also, included the testing specifications developed to verify the SPARK design by White-box and Black-box testing procedures. The overall user acceptance of the SPARK is measured by independent usability survey carried out based of System Usability Scale model and the prospective user input based on their experience is discussed. In addition, system performance has been monitored by analysing real time GWAS dataset.

Finally, this chapter summarises the SPARK functionality with the genome browsers and analytical platforms as the overall software platform for the GWAS. The detailed discussion of each platform was made in the literature review chapter and a comparison is made in this chapter (Table 7.2).

7.2 Evaluation of software before introduction to a work domain

Software systems and packages are built to consider a specific question and provide a related application-based approach to solve the problem. A proposed system can be applied to a fresh venue for a new system, migrated from an existing system, or extra functionality added to an existing system [216]. The risk of success or failure after introducing a software system needs to be considered. As a precaution to

avoid such mishaps, software systems should be evaluated.

Introducing a new system to a work environment creates excitement with new work patterns and panic with the possibility of failure which can lead to ongoing work activities being neglected [217]. The excitement usually drives the workforce to learn and experiment with the system functionality. Therefore, a carefully developed training approach and documentation for the system functionality is needed. It is important to encourage the users to participate in the training programs and refer to the system functionality documentation. The close collaboration with the developers and user community enables a smooth transformation of work patterns towards a more automated approach. Despite the increased levels of excitement, there is a huge risk of failure in the work environment due to a misunderstanding of the functionality and system-specific technical issues. Also, the work community may be resistant to adapt to new work patterns and take on extra responsibilities to interact with the automated environment.

Migration from an existing system is typical in technology-driven information system workspaces [218]. There are various reasons for replacing an existing system with a new system, including adopting the benefits of recent information technology innovations (e.g. moving desktop-based systems to web-based information systems), the end of the existing systems support period, or management making the decision to introduce a new system. Comparison of the previous system with the new system is likely among the users and expectations of the new system may be high. There may be high resistance from the user community with replacement systems which will require a higher level of adaption process. Compared to the first approach of introducing a new system to the work environment, the replacement requires training materials which consider the user's knowledge of the previous system, dissemination of information about data migration process, and more importantly, consideration of the psychological biases of the user to the new system. A carefully designed data migration and training program are capable of gaining user acceptance for a new system based on their merit.

Adding extra functionality to an existing system is a typical requirement when adapting to the modern business use case scenarios [219]. Modifications accepted by the current system need to be integrated smoothly, changes should not interrupt the existing functionality, the existing datasets reside in the system are not violated, and there are value-added benefits to the users. Rather than introducing a whole new system or replacing an existing system, additional functionality does not require system-wise training or psychological acceptance of existing users. A highly tested piece of software integrated with the existing system would be required and the user skills updated to interact with new module or function.

The evaluation of a software system process is mainly considered for enterprise management information systems which are more prominent at an organisation level with high impact on data-related operations over a number of years. The evaluation process is important to validate the system acceptability among the users and cost-benefit comparison of the system investment. Following a number of enterprise management information systems and managers' feedback about these systems, a number of factors which impact on enterprise systems have been identified [220]. These include (1) required functionality of the system, (2) the system reliability, (3) the total cost incurred for the system, (4) ease of customisation, (5) ease of use system in an operational environment, (6) the software provider reputation, and (7) ease of software implementation.

Evaluation of SPARK has been based on the decision-making points chosen for an enterprise system. Choosing enterprise information system evaluation criteria for the SPARK system is based on its design specification (see Chapter 4). SPARK system design is more aligned with a management information system design rather than the individual bioinformatics analysing tool based on The Ark platform. Furthermore, PLINK and BLAST bioinformatics tools (see Section 2.4.2) have been compared with SPARK. A GWAS data management facility has been provided rather than an analysis tool. The SPARK system has included GWAS dataset management, search on GWAS dataset properties, managing the analytical

packages, and executing the analysis and reporting the results (see Section 5.4). The functionality mentioned above can be compared to the tasks developed in a management information system which manages multiple organisation activities by recording each individual event as a record. The difference is that SPARK recorded the meta attributes rather than their implications. Based on the meta information recorded, SPARK will carry out the GWAS analysis according to the specified details. For example, to execute an analysis the Micro Service meta attribute needs to be chosen to initialise a connection, the data source and computational package attribute to select dataset and analysis package to reside in the execution environment. The analysis meta attributes have been recorded in the SPARK system to monitor the status and extract the results.

The SPARK evaluation has been carried out based on two existing GWAS analysis systems which have a proven track record in the research community. Therefore, Galaxy and SeqWare (see Section 2.2.3) platforms were selected as the benchmarking systems to compare the SPARK functionality based on the information system evaluation factors.

7.3 SPARK-specific evaluation criteria

Based on the seven principals of enterprise system evaluation criteria (see Section 7.2), the SPARK system has been evaluated by comparing its functionality with two other leading open-source software systems for the GWAS researchers, Galaxy and SeqWare.

7.3.1 SPARK system functionality

The SPARK project aims to develop an application platform which supports GWAS data management and analysis via existing biomedical data management system (see Section 5.2). The design objectives of SPARK were driven by the existing

platforms to browse and perform analysis on genomic data, as discussed earlier in the thesis (see Section 5.2) and integrated with an existing web-based study data management environment. Three entities of the GWAS analysis platform were identified which consisted of data management, analysis, and sharing the results. The Ark genomics module facilitates a common web user interface for the distributed SPARK instances which is operated remotely. The Ark genomics module operates using the pre-established and comprehensive security best practices of The Ark. However, SPARK is not tightly coupled to The Ark; it operates independently of The Ark by way of secure communication channels (web services, see Chapter 5). The SPARK module of The Ark, dubbed the "Genomics Module", comprises the following four major components:

Micro Service:

Service locators to communicate with the SPARK nodes via the web services and facilitate communication between The Ark, SPARK node, database storage engine and analysis platform.

Data Centre:

Permitting researchers to navigate, manage and query the genomic datasets which reside in potentially large-scale storages that are linked to SPARK service nodes via the Micro Service implementation.

Computation:

Computation packages consist of all materials, e.g. source code, compilation "make" files (scripts), reference data, etc., to deploy a GWAS analysis algorithm on a computational facility. The "plug-in" style packaging of SPARK's computation packages promotes sharing of the implementation among research groups.

Analysis:

A web-based interface to run an analysis algorithm (deployed as a Computation package) on a selected dataset (visible via Data Centre connections) and store the results returned.

Compared with SeqWare and Galaxy platforms, the functionality of SPARK is superior. The SPARK design tends to operate as a GWAS data management platform and analysis executor. SeqWare (see Section 2.2.3) focusses on workspace management with multiple steps of a GWAS analysis for a specific dataset including, data cleaning, executing multiple analysis techniques, and result extraction, and also managing the execution workspace connected to cloud computing infrastructure. In addition, there is a meta database consisting of the dataset properties which is updated according to the analysis dataset. Researchers communicate with the SeqWare system using the web interface and standalone common line client applications.

The Galaxy platform (see Section 2.4.2) is deviated by the pathway introduced by the SeqWare system. Galaxy focusses on individual datasets and executing the analysis on top of a dataset. The analysis methods are predefined, and the execution results will be either reuse for another analysis or input to a visualisation method predefined in the system. Therefore, Galaxy is able to run multiple analyses on the same dataset and filter the analysis results based on existing filtering criteria. Galaxy is a web-based system consisting of plug-in management to allow developing additional methods to analyse and visualise the GWAS datasets.

The SPARK design is more effective than the Galaxy system and SeqWare. The SPARK design has one advantage over these systems due to its independent implementation environment. Researchers can connect to multiple execution environments using a single web user interface and carry out the GWAS analysis within a single study space. Also, GWAS researchers can store multiple computational packages suitable for different high-performance computing environments and execute these computing packages from different data sources. The web service-based

distributed nature of SPARK design has enabled more independent execution of scenarios distributed on various infrastructures without making a single job queue.

7.3.2 SPARK system reliability evaluation

The reliability of the results generated by a software system is a key evaluation factor of its quality and acceptability. Therefore, comparing the existing GWAS systems reliability with the SPARK system is important to discuss, in particular with the SeqWare and Galaxy systems. The GWAS analysis techniques implemented to manage and analyse specifically to the existing systems are proven and well-tested before using a study. In addition, system-wise existing techniques have established much confidence in the researchers.

The SPARK deviates from the existing approach of Galaxy and SeqWare systems. As discussed in Section 7.3.1, SPARK is connected to distributed services and is compliant with a SPARK-specific application programming interface (API). Therefore, data management mechanisms (GWAS data search and querying mechanisms) and analysing techniques (Computational packages) rely on the original author's accuracy. In addition, SPARK provides the design reliability of existing implementation of The Ark system based on micro service management, computational package storage, and monitoring the process statuses.

Implementation of The Ark system provides confidence in the reliability of the functionality (see Chapter 3). Based on the original motivations of the SPARK design to manage the distributed GWAS datasets and analysis, reliability is a much debatable constraint. As a data management system compliant with the service management, SPARK has a proven track record of a secure and reliable data storage engine based on The Ark system. The GWAS data storage and analysis wise it has been designed to rely on the service providers who host the SPARK nodes and are compliant with the independent reliability standards.

7.3.3 The total cost incurred for initiating SPARK system

The cost of a system is a crucial factor when choosing a software platform for the operational environment. There have been studies conducted to measure the impact of cost factors of choosing an enterprise resource planning (ERP) system [221], but little study has been conducted to check the cost impact of selecting a biomedical research platform. However, the factors identified in ERP systems can be applied to other systems by the nature of the facts identified in the process. Therefore, a SPARK design cost-based evaluation has been conducted based on the factors considered in ERP and compared with Galaxy and SeqWare systems.

The cost of a software system is comprised of direct and indirect factors. The direct costs include the purchasing cost of a software platform and any other costs related to the acquisition process. The direct costs are visible and easily measured as tangible figures, the indirect cost is more difficult to determine and may deviate from the domain-specific requirements. SPARK, Galaxy, and SeqWare system are open-source based and require zero cost to acquire. However, the complexity of the system functionalities and initial configurations require many hours of work to make them functional, hence increasing the indirect cost.

In particular, SeqWare and Galaxy systems require extensible configurations on an initial set-up by trained computer specialists. In addition to setting up the platforms, researchers have to be trained extensively on the available functionality. A significant time investment in the training process will be required, but the advantage is the researchers are a specialised set of users who are eager to learn new things. However, breaking the usual practices with standalone tools developed in the bioinformatics centres would be crucial and to prove the superiority of the external functionality of Galaxy and SeqWare systems will require several test analyses to be run.

The SPARK design has less indirect costs compared with Galaxy and SeqWare systems due to the nature of the design. SPARK installation costs will be minimal

due to availability of The Ark system and distributed SPARK node deployment by specialised computer scientists attached to computing facilities. The Ark system will be introduced to the researchers before moving to GWAS data management. Therefore, the cost incurred in learning the system functionality would be minimal based on the researcher's prior experience with The Ark functionality. The changes to the analytical methods will occur at the computational package level without affecting the system upgrade cost or requiring additional testing. The system upgrades will be compliant with open-source base system upgrades based on The Ark open-source community.

7.3.4 The SPARK eases of customisation

Customisation of a software system is an important factor in the usability evaluation. Process customisation has referred to the system compliance of adopting the changes by user demand [220]. Therefore, software design has given significant weight to the system's customisability and high availability of changes after modification (see Section 5.5).

The design of the SeqWare and Galaxy systems enables customisations. The SeqWare client-server-based design provides the freedom to customise the server side features without affecting the clients. In a scenario of deploying the SeqWare inside a server explicit to its preferable Amazon S3 servers, there would be a facility to change the deployment settings. Also, the meta information-based database is compliant with the changes suited for the datasets which consists of unique meta information sets based on GWAS. The analysis pipelines can be developed independently and executed on a selected dataset separately.

Also, the Galaxy system is deployed independently and enables the web user interface-based customisation mechanism to apply the changes to analysis. The web interface of the Galaxy system has been developed to choose the pre-configured analysing method for a dataset. Web-based controls configure the input parameters

to the analysis methods to control the execution based on the researcher requirements. The level of the customisability enables the system to cater for the diverse GWAS research requirements made by the researchers. Also, Galaxy provides an external programming interface to develop the researcher-specific analytical and visualisation methods which are embedded in the system.

SPARK design also highlights the customisability of the GWAS research requirements as a medical research platform. The data centre design enabled on-demand changes to GWAS data management by enabling distributed data services. In a case of enabling new data storage with unique access criteria, modification of the connecting SPARK node would be needed and would not interrupt the data operations enable with other services. According to the GWAS data format, data searching and querying mechanisms can be modified and changes will be immediately visible to the researchers via The Ark genomic module.

Similar to the data centre customisability, analysing methods can be rapidly changed via the SPARK design. The analysis methods reside as computational packages attached to The Ark genomics module and the most recent computation package with the required changes can be uploaded. Therefore, changing an analysing method does not require a system or module upgrade. The results can be compared by parallel execution of analysis methods with different versions and the results of the analysis methods evaluated.

The SPARK-based GWAS data management and analysis are prone to customisation and are not interrupted by changes by distributed service node deployment. Customisation would be challenging only if required to change the existing Ark genomic programming interface and new specifications introduced. In that case, all the connecting SPARK nodes would need to be modified.

7.3.5 Ease of use system in operational environment

System usability is an important factor in an operational environment [210]. The usability aspect of a software design and the importance of considering usability approach adopted to the SPARK design has been discussed in the SPARK chapter (see Section 5.5.7). This section looks into the usability aspect from a software user's point of view.

In the usability evaluation process, the acceptability of the software users and how they adapt to the particular software package should be considered, including analysis of the psychological acceptance of selected software system, how much cost benefit has been incurred after introducing the software system to the operational environment, and how fast the users have adapted to the workflows introduced by the software system.

Considering the Galaxy and SeqWare software platforms, both systems have gained researcher acceptability in GWAS studies. Researchers have introduced these systems to students as preferred platforms and have developed training manual for guidance. These systems are the first preference of the GWAS researchers who have hands-on experience with the systems. Also, the feedback opted from the GWAS researchers for a certain period has helped to increase the usability of the Galaxy and SeqWare systems by improving the web user interface to manage the analytical aspects through carefully designed GWAS-specific web controllers and fixing the reported programming faults by researchers.

The SPARK design is a novel approach and developed as a PhD research project which did not have a large GWAS researcher user base and timeline of multiple software release versions. The Ark is a mature software platform (see Chapter 3) and has evolved during more than a decade of the software development project. The SPARK systems web controllers are implemented based on The Ark platform and follow the functional workflows of the existing system. The prospective user base for SPARK system is the existing Ark users, who will be interacting with the system via The Ark genomics module. The technical design is hidden from

researchers based on the original objective of SPARK design (see Section 5.2) to protect the GWAS researchers from the computational complexities of the system.

7.3.6 The software provider reputation of the SPARK

The software provider's reputation is a much-considered factor in the software acquisition process. The reputation of the software developer is dominated by the track record of the previous software implementations, the time it took to respond to a software issue, and the user's perspective of the software vendor. Therefore, a decision on a software platform for an operational environment may potentially have a large impact on a software provider's reputation.

SeqWare and Galaxy are platforms based on academic backgrounds which were developed as collaborative work between researchers attached to prestigious universities, including UCLA, Penn State, John Hopkins and Washington. Therefore, they inherently gained a good reputation for these systems through their collaborative institutes and established a track record of successfully conducting the biomedical research based on these systems. Also, a large number of academic citations (>1000) referring to these systems have been made, according to the Google Scholar citation engine.

The SPARK project is based on The Ark system which is accredited by The University of Melbourne and The University of Western Australia epidemiological research community. The Ark system has been extensively evaluated and used for multiple epidemiological research study data management projects. The developers who collaborated for The Ark project have been involved in the SPARK project. Therefore, the SPARK system provides much assurance due to its vendor reputation and is open to collaborative parties to contribute as an open-source project.

7.3.7 Ease of software implementation of SPARK system

The software implementation is a process designed to deliver virtual appliance capable of accomplishing specific functionality accurately and reliably in the acceptable time frame. Therefore, software implementation time is a critical factor to be considered by evaluating the ease of implementation [216]. The time taken to add new functionality or to fix a software issue has to be minimal to consider if the software is suitable. Good software design and management require qualities, including security, modifiability, availability, etc.(see Section 5.5), and enable less developer coding, testing, and deployment activities to deliver the change to the user.

A large community of open-source software developers have been contributed to the Galaxy and SeqWare systems. Therefore, addressing an issue or improvement receives maximum attention by the developer community based on the impotence of the task. Also, given platforms come with well-established programming interfaces and much-needed documentation is commonly available on the web. Each change made by the developer community is bound to a major release of an application platform because the systems are operated as a single instance and consider the current executing tasks. Also, one change can impact the existing methods developed by the collaborators and should be tested against all the possible scenarios affecting the change. Therefore, before applying the change, halting the server of the selected applications is required.

The SPARK development community is relatively small compared to the SeqWare and Galaxy platforms. Similar to the compared platforms, SPARK design is based on The Ark programming interfaces and supported by the open-source community, therefore experienced developers are able to contribute to the source code and project tasks. Compared with the selected systems; there is one major advantage inherent to the implementation process of the SPARK distributed design. Web interface-related changes are only attached to The Ark deployment, but SPARK nodes are managed independently by facility owners. Therefore, in the distributed SPARK a change or solution developed to fix an issue only affects itself

rather than halting the entire system.

In addition, data management, computational package management and hosting, and analysis execution tasks are independently operated by SPARK nodes. Specialised developers are assigned by the facility management dedicated to the SPARK node and computation package development. The computational packages which are always implemented and tested by one developer group are available for the study collaborators rather re-implementing that analysis themselves. The distributed design helps to maintain The Ark genomic module source code by the core development team and SPARK node source code by the facility ownership.

7.4 SPARK system evaluation

Based on the seven objectives introduced in the previous section (See Section 7.3), The SPARK system design and implementation is evaluated. The functional testing was chosen to evaluate the reliability of the system functionality, and user acceptance testing is chosen to evaluate the extent the prospective user community is aligned with the system design and functionality.

7.4.1 SPARK system testing

SPARK system testing is carried out based on known software testing approaches called White-box and Black-box testing. In the White-box testing evaluation, the initial software design and quality of the code embedded in to the SPARK implementation was assessed. The SPARK design and architecture was elaborated in the Chapter 5 and justifications are provided for choosing the microservice based software architecture.

The next part of the White-box testing is evaluating the code quality. The current SPARK source code is evaluated using the well-established code language static analyser called PMD [222]. It has been developed to highlight the static

programming flaws including unused variables, infinite loops, non-caught exceptions etc [223]. Therefore, SPARK source code has been successfully checked against the 130 Static rule set defined in the PMD and those rules have been described in the PMD documentation [224].

The Black-box testing procedures covered a set of well-defined functional test cases targeted for 4 major SPARK components. Mainly interacting with The Ark genomics module Micro Service, Data Centre, Computation, and Analysis tabs, test cases were executed and results were compared with the expected outcomes. The functional test cases are specified in the Appendix D with the input and expected output. The test included negative and positive test cases to cross check the functionality of the system.

7.4.2 SPARK usability testing

To evaluate the usability of the SPARK design, a proper system usability survey has been carried out among the prospective users of the system. The survey included experienced GWAS researchers, software engineers, data managers, biostatisticians, and software architects who possess background knowledge in computational biology.

To the survey participants, overview documentation of the SPARK design and user manual to carry out a sample GWAS analysis within feasible time frame (See Appendix B) were provided. The survey included two types of question groups (See Appendix C). First question group targeted to evaluate the spark design in the context of general system usability scale [225] and second group targeted evaluating the SPARK specific functionality. In the end of the survey, users were requested to provide feedback on the survey approach and the system.

Responses from 12 participants were retrieved for the survey and the following results summarise the survey outcome. Summarisation is done based on Likert scale which is a proven technique to evaluate the user acceptance of a system [226]. The

group 1 responses which are based on the system usability scale are summarised and presented as a box plot in Figure 7.1.

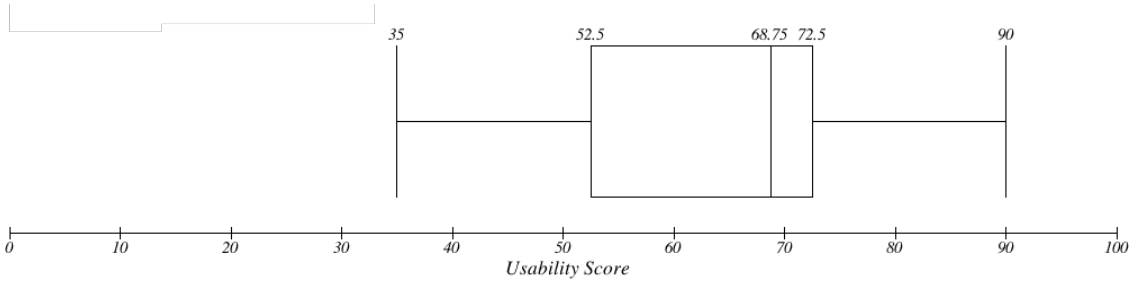


Figure 7.1: SPARK – System Usability Score Distribution

Based on Figure 7.1, system usability scores are situated between 35 minimum and 90 maximum scales inside the overall score ranging from 0 to 100. There, 52.5 represents the lower percentile and 72.5 represents the upper percentile of the system usability scores. As a summary result, 68.75 median value represents the overall usability score of the system based on participant experience.

The second question group tested the understandability of major components in the SPARK system (See Section 5.4). Therefore, the user experience in essential functionality of the SPARK design related to Micro Service, Data Centre, Computation, and Analysis components are evaluated. The questions directed to the users in the second part of the survey as follows.

- Q1. It is easy to configure a new micro service?
- Q2. The status of a micro service is clearly indicated and understood?
- Q3. I understand the meaning of the online/ offline status for a dataset?
- Q4. It is easy to navigate to a specific data source?
- Q5. I am confident in extracting genotype records using the query builder?
- Q6. I understand how to record meta-information for Genome-Wide Association Study (GWAS) data.

- Q7. I understand the meaning of the uploaded/installed status of a computational package.
- Q8. I understand how to install a new computational package
- Q9. I am able to extract the results of an analysis.
- Q10. I am confident in performing analyses using the available computational packages and data sources.

Based on the answers received for the questions from the second part of the survey, median scores were calculated for each question and Figure 7.2 depicts those scores. Figure 7.2 has clearly indicated that median scores obtained for each SPARK functionality related question are positioned in the acceptable regions (3.5 and 4) according to the Likert scale.

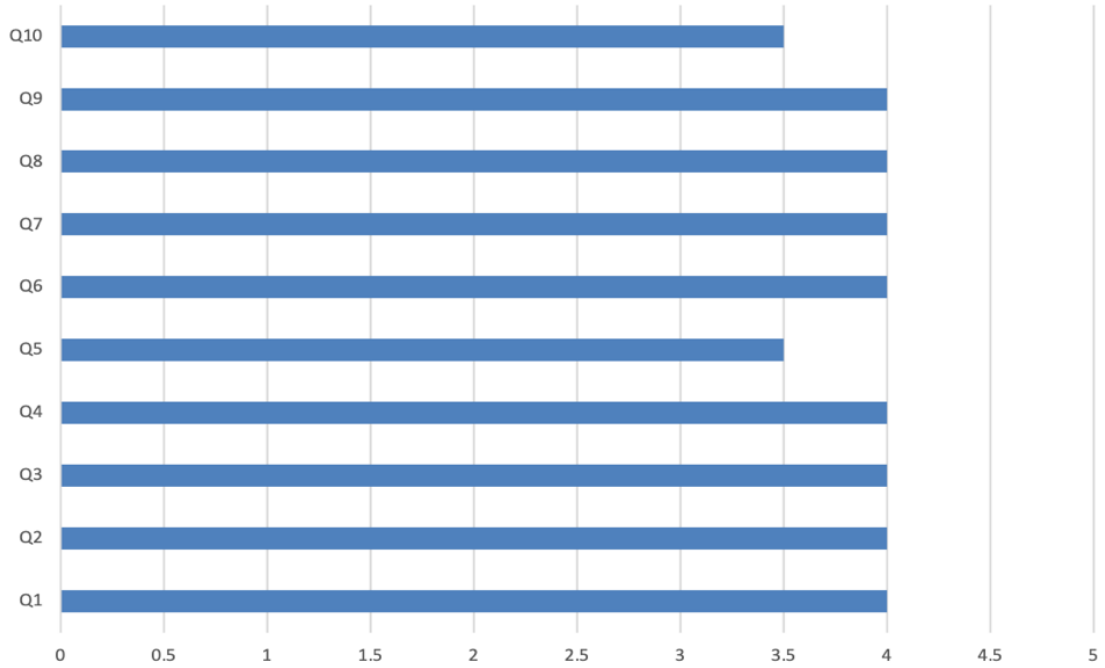


Figure 7.2: SPARK functionality evaluation score based on the user input

The final part of the usability evaluation survey is to receive the user comments based on their experience and suggestions to improve the existing functionality.

One major requirement came across by the usability survey is the lack of options to monitor the execution status. There, users have suggested providing a more dynamic web panel to monitor progress of a process and real time status change when process is completed.

Another suggestion for improvement mentioned by the users were introducing a log record per analysis to monitor the execution steps and the status. Also, other suggestions included introducing a dynamically manageable analysis with logical workflows and enabling automatic inputs where one analysis output becomes the input for another analysis. Implementing those changes will further enhance the existing functionality defined in the SPARK system with enhancements to The Ark module architecture as well.

7.4.3 SPARK performance evaluation

For the testing and demonstration purpose, SPARK instance is hosted in the Nectar cloud. There, SPARK node is configured with 64 GB RAM based 16 virtual central processing units (vCPU) based virtual machine. Given set-up ties up with the original objectives of the SPARK design on easily integrating with high-performance computing sources and distributed data centers. Furthermore, above set-up is pre-configured and independent of the user knowledge on system configurations.

For the performance evaluation, original GWAS dataset which includes 16207258 SNP variants has been utilised. Relevant dataset was derived from the Breast Cancer Association Consortium custom SNP genotyping array which includes SNP variants from 830 Female participants [227]. For the above dataset PLINK based analysis were conducted via the SPARK node by purely interacting with The Ark Genomic module web interfaces. The analysis execution times were summarized in Table 7.1.

Analysis Type	Execution Time (hh:mm:ss)
Summary statistics analysis for missing rates	00:21:08
Summary statistic analysis for allele frequencies	00:07:03
Basic Association Analysis	00:10:04
Sorted Basic Association Analysis results includes a range of significance values adjusted for multiple testing	00:13:06

Table 7.1: Analysis execution times for an original GWAS dataset

Therefore, based on this experiment, conducting custom GWAS analysis utilising GWAS analysis tools provided by the SPARK node is evident. Furthermore, results can be derived by the researchers within a justifiable time-frame.

7.5 An overall comparison of the SPARK functionality with existing genomic management systems

Compared with the genomic analysis platforms, the SPARK project provides a secure workspace for the analysis execution and results sharing. Analysis results are not limited to the researcher who initiated an analysis for a study, but are available to all researchers who have access to that study within The Ark.

Using a combination of Microservice, Data centre, Computation, and Analysis architectural components, SPARK offers a user-friendly web interface to the GWAS researchers in a manner that is integrated with a secure biomedical data management environment. A comparison of The Ark genomics module's capabilities with leading genome browsers and analytical frameworks is summarised in Table 7.2.

According to the table summary, the SPARK project has demonstrated its capabilities and design perspectives against the known genome browsers and genomic analytical platforms. SPARK has matched its capabilities with the existing genomic platforms and is open to further enhance its capabilities.

Feature	SPARK	Genome Browser		Analytical Frameworks	
		Ensemble browser	UCSC browser	SeqWare	Galaxy
Study-based genomic data management	Yes	No	No	No	Yes
Support for high-performance computing platforms	Yes	Yes	Yes	Yes	Yes
Heterogeneous genomic data management	Yes	Yes	Yes	Yes	Yes
Operated by third-party hardware infrastructure	Yes	Yes	Yes	Yes	Yes
Open-source licence	Yes	Yes	No	Yes	Yes
Genomic data visualisation	No	Yes	Yes	No	Yes
In-built phenotypic data management capabilities	Yes	No	No	No	No
Secure dataset sharing	Yes	No	No	No	Yes
Ability to integrate third-party tools	Yes	Yes	No	Yes	Yes
Genomic analysis pipeline	Yes	No	No	Yes	Yes
Application programming interface support	Yes	Yes	Yes	Yes	Yes

Table 7.2: A capabilities comparison for SPARK, genome browsers and analytical platforms.

7.6 Summary

This chapter discussed the SPARK system and its suitability as a default GWAS data management system. The applicability of SPARK system as the preferred analytical platform in the research environment and its adaptability to utilise the research outcomes was discussed. The evaluation is carried out based on the predefined enterprise system evaluation attributes and the SPARK system satisfies all criteria associated with an enterprise platform operating in a research environment.

In Section 7.2, the individual enterprise system evaluation attributes addressed by the SPARK, SeqWare, and Galaxy systems in GWAS data management were discussed. The SPARK micro service, data centre, computation, and analysis provided much-needed operational support to conduct a GWAS analysis based on functionality, reliability, cost, customisability, adaptability, vendor reputation, and ease of implementation factors. These attributes were discussed in relation to the

SeqWare and Galaxy platforms and a comparison of each attribute was made for the systems.

The section 7.4 presented details on testing specifications developed to test the SPARK design by conducting White-box and Black-box testing procedures. Positive remarks were received to the initial SPARK design by the System Usability Scale model and the prospective user feedback provided much clear path to improve the design of the SPARK system to cater for wider researcher community. Also system has been tested with an original GWAS dataset and performance indicated the system's ability to handle large datasets.

Finally, Section 7.5 has discussed the overall GWAS research management criteria against SPARK, genome browsers, and analytical platforms. The initial SPARK design has satisfied much-needed GWAS operational and management requirements compared with other prominent genomic data management and analytical platforms. Therefore, SPARK is a functionally ready system which is capable of managing a GWAS operation environment based on a biomedical study space.

The SPARK approach to enable a novel data management platform for biomedical research datasets and the future research directions established through this work.

8 Conclusion

8.1 SPARK – A new platform for genomic research

This thesis has contributed to the advancement of architectural design and the development of best practices for a next-generation biomedical data management system. The Ark system (see Chapter 3) was chosen as a prominent biomedical data management system and demonstrated the architectural design enhancements based on the existing implementation. The SPARK project research focus was mainly targeted on how to manage the pedigree and GWAS datasets inside existing study-based biomedical research data management system.

The aims of the SPARK project were formulated as a set of four research ob-

jectives. The contributions of this thesis will now be retrospectively summarised in regard to these research objectives. In this way, the significance and novelty of the SPARK project will be highlighted.

Pedigree data management

In the past decade, biomedical data management systems have evolved on many fronts. Their original data management objectives have diversified, and a new set of requirements have emerged. The main reasons for these changes include recent advances in information systems and supporting frameworks [76]. The Ark was developed based on the first wave of biomedical data management systems and plunged into the study-based biomedical data management aspects (see Section 3.2).

The Ark chapter (see Chapter 3) has provided a detailed description of The Ark's architecture and its modular design. The modular architecture has helped The Ark's distributed development environment and its rapid development phase. The Ark's role-based security infrastructure has provided a secure environment for biomedical datasets. It gives system administrators the ability to restrict access levels for users based on their roles. The Ark's capabilities were then compared with other biomedical data management systems. There has not been an existing biomedical data management system capable of managing the pedigree datasets within the study context (see Section 3.13). This feature shows the importance of using The Ark for biomedical data management.

Chapter 4 contributes a new pedigree data management module to The Ark. A data model was developed to operate in conjunction with The Ark's approach of first bringing a study into context, and then bringing a subject within that study into context. The new data schema simplifies the problem of modelling potentially complex family structures, requiring only the specification of parental and twin relationships for each subject (see Section 4.4). This mechanism has developed the

BloodLine algorithm to infer and label the relatives dynamically for a given subject (see Section 4.6). Representation of pedigree data in a graph data structure enabled detection of consanguineous relationships using circular validations (see Section 4.7) and provided validation mechanisms to detect the consanguineous relationships in pedigree datasets. Specialised Java program code was developed to integrate Madeline [181], facilitating visualisations of pedigree structures inferred by the BloodLine algorithm. Bulk uploading enables large family dataset to be imported to The Ark with validation of consanguineous relationships.

Together, these features and capabilities mean that the pedigree module provides a complete pedigree data management solution for researchers handling large family datasets and enables them to access the optimum existing study-based data management system functionality. Therefore, the thesis has contributed to solving a major limitation of the existing biomedical data management system approach by managing the pedigree datasets within a study schema-based database and enabling an eternalised visualisation mechanism for automatically chosen pedigree datasets based on the selected subject. Finally, researchers do not need to rely on the data managers to deal with the pedigree datasets, enabling them to manage a pedigree dataset via the web-based data management system.

Access to high-performance computing resources

Access to HPC sources have increased with the advancements in the genomics research domain (see Section 2.4.1), the emergence of the Big Data management systems (see Section 2.2.3), and the emergence of personal genomics (see Section 2.4.4). The introduction of the biomedical informatics domain was based on the need to apply information science to the medical research domain. Genomic research has been tremendously uplifted after completing the HGP [228], including the sequencing of a number of human genome samples based on various human origins such as geographical discrepancies, ethnicities, etc. To find out the common causality

of genomic alleles applied to various phenotypic identities has been investigated by GWAS. There are various analytical techniques developed to understand the genetic variation regarding phenotypic items, and systems have been developed to integrate analysing techniques to multiple data sources. The nature of the existing genomic datasets has required improved Big Data management techniques and HPC facilities to support the heterogeneous complex genomic analysis.

The next phase of the biomedical data management systems is to integrate the genomic data management and analysis with the rest of the system. The literature review discussed the various forms of HPC facilities (supercomputers, cluster computing, grid computing and cloud computing) and specialised techniques developed to utilise their computing power (see Section 2.5.1). Chapter 5 has contributed an architectural software design which enables collaborative genomic data management and analysis in a manner that integrates with The Ark. The available software architectural designs (monolithic, N-tier and Service Oriented Architecture) and their compatibility with the SPARK microservice architecture design are summarised. A novel microservice architectural approach was developed, allowing one instance of The Ark to integrate with multiple SPARK nodes in a distributed manner. The SPARK architecture presented in Chapter 5 could also be used as a basis to extend to other biomedical data management systems such as REDCap [31].

Enabling software architecture for an existing software biomedical data management system to interact with HPC facility provides much needed analysing power for datasets managed by the systems. Rather than extracting the datasets to manually move to HPC facilities, SPARK design enabled the inherent mechanism to interact with facilities via a web-based interface (see Section 5.4). Also, microservice architecture has provided an independent design for the SPARK nodes which have unique constraints to accommodate their HPC system.

These architectural goals have been satisfied by the SPARK microservice architecture design. The proposed architecture has enabled the multi-stage secure

design to access the HPC facilities and data storage facilities, which are unique to each facility access specifications (see Section 5.5.1). The service availability is maximised by the independent services defined for each SPARK node operating independently and managed by single web interface by selecting an available service centre (see Section 5.5.2). The system interoperability is increased by independent SPARK nodes and communication via the REST-based web services (see Section 5.5.3). The modifiability of the SPARK design is maximised by microservice definition-based operations independent from each other and computational package definitions with unique operational instructions suited for each execution facility (see Section 5.5.4). This design has increased the performance by not being limited to a single HPC facility but enabling parallel execution of multiple analyses via distributed computing facilities which were initiated with individual SPARK nodes (see Section 5.5.5).

Similar to performance attributes, SPARK design has increased the testability of the system via individual services dedicated for each test scenarios supported by mock service objects (see Section

Genomic data management

Genomic data management is a key aspect of SPARK design. The objective is to introduce a relational-based and non-relation-based mix approach to manage the GWAS datasets and enable preconfigured extraction mechanisms to search in genomic datasets. According to the literature review, the data management approach has played a major role in the informatics systems and introduced a number of data models suitable for domain-specific requirements (see Section 2.2.1). The relational model was the most accepted data model in many data management systems, and multiple datasets relied on the relational model. There have been limitations in relational data model, including the preconfigured data schemas, database scale up with the large datasets residing in distributed operational environments, the cost of

extreme time consumption to extract information from large datasets, and physical disk space management in storing schema-based datasets (see Section 2.2.2).

Motivated by this objective, Chapter 6 has evaluated conventional, relational database approaches and non-database representations (i.e. files on disk) using disk space and data loading times as evaluation metrics. A database model is not feasible if it cannot perform a basic operation such as data import within a practical time frame. The results of the experiments show that relational database systems are impractical for GWAS data management; the time required for fundamental operations, such as dataset insertion, grows into the order of hours and days for typical GWAS datasets (see Section 6.6). Also, physical disk storage allocation in relational data schema is extremely high compared to GWAS datasets file storage. As a result, alternative data storage approaches based on non-relational models (NoSQL [62]) were investigated. The popular non-relational data models were considered, including key-value, document, columnar and graph models for an efficient data model for GWAS datasets.

The non-relational columnar database called Cassandra [229] was selected for integration with SPARK because the columnar data model was shown to perform well; both regarding speed and dataset size-on-disk. Using the Cassandra columnar database within SPARK, GWAS data is represented using an efficient binary representation in which the data for 16 alleles is encoded in a single column, using one byte of storage (see Section 6.8). In this way, the tailored use of Cassandra addressed the problem of efficient data storage. However, for the database to be useful to researchers, an additional software layer was necessary to facilitate common operations on the database contents, e.g. querying, extraction of data subsets, and decoding of the contents of columns. The Lambda architecture [50] was used to implement these operations, which can be lengthy, and return immediate control of the SPARK user interface to the researcher.

This architecture has shown promising results for the GWAS datasets which can be further optimised to suit other genomic datasets. Genomic datasets con-

tain many of the repeating elements and similar encoding proposed for GWAS can be implemented for other genomic datasets too. Extra data layers in application development can ease the technical difficulties of maintaining matching data representation of the table structure by following the Lambda architecture for application data tiers in other genomic data management systems.

Fostering biomedical research collaboration

After the first wave, the GWAS suggested creating more collaborative research by identifying significant disease causalities through various studies [46]. Much closer collaboration between the GWAS research groups working across the globe can be achieved by sharing the existing datasets, novel GWAS analysing algorithms developed individually, sharing the results to avoid repeating the same experiment, and sharing expensive HPC power. The World Wide Web is a common ground for this approach, yet there have been multiple challenges in this space, including the privacy, ethics approval and mutual understanding between the research groups.

The existing biomedical data management system has provided a web-based collaborative data management mechanism for biomedical datasets (e.g. The Ark and REDCap). These systems focused on managing the phenotypic datasets belonging to the biomedical research groups. The challenge of managing the genomic datasets has been dedicated to individual systems belonging to research groups and operated based on unique computing hardware which supported these systems. SeqWare [70] and Galaxy [45] systems provide a unique operational environment for genomic data management and analysing based on their deployment environment.

SPARK simplifies the complicated processes of GWAS data management and analysis in a user-friendly, study-based workspace. Based on The Ark system (see Chapter 3) and compliant with its secure data management mechanism, SPARK provides a much needed link between the phenotypic and genotypic data management by a single web-based system. Four major sections support the GWAS

researchers to avoid the overhead of computational practice and experience in interacting with HPC sources and large GWAS datasets in research work. The microservice section would simplify the connection to an analysing platform to a single web with its availability indicated by the status flag (see Section 5.4.1). The data centre section represents the available data storage related to a SPARK node indicated by the microservices and represents a virtual map of files residing in this storage (see Section 5.4.2). In addition, query capabilities to the PLINK compatible datasets reside in the data centre based on predefined query functions. The computation section represents the SPARK-specific computational packages repository and deploys these packages to selected computing facilities identified by the microservice (see Section 5.4.3). The analysing section is the web-based workspace to carry out the GWAS analysis based on the available dataset. During the analysis, deployed computation package in a selected micro service will be applied to the dataset (see Section 5.4.4). The completed analysis results can be downloaded later and analysed by the third party tools. The given design enables researchers to work inside a study workspace predefined by The Ark system, respecting its security constraints (see Section 5.5.1).

Chapter 7 compares the features and limitations of SPARK concerning popular genomic data management platforms, including SeqWare and Galaxy. The available facilities in the genomics browsers, the genomic data import/export capabilities available in the browsers, and provision to customise these browsers for individual research objectives are discussed (see Section 7.5). Similar to the genomic browsers, the chapter discusses design and capabilities of analytical platforms available for GWAS. The simplified design approach followed by the SPARK project to enable study-based GWAS research management by microservices, data centres, computation and analysis facilities are discussed in the SPARK implementation. The simplified web user interface provided by The Ark genomics module avoids the burden of exposing the GWAS researchers to complex system implementation of Big Data management powered by HPC sources.

The objectives elaborated earlier have provided a roadmap of the SPARK project and how it has been progressed as a research study. The experience and results derived from the SPARK research has led to a set of future research perspectives. There would be a number of design perspectives generated by studying the workaround behaviours of the initial SPARK design. Those designs perspectives have been initiated based on satisfying the present day GWAS study focus towards the commercial and personal genomics successes. In the future work section, identified research perspectives based on the original SPARK design and development will be discussed.

8.2 Future work

While the SPARK project has made substantial contributions towards a researcher-friendly platform for genomic data management and HPC-enabled analyses, numerous avenues of future work remain. The architectural design and open-source nature of SPARK mean that members of the global bioinformatics and computer science communities can readily contribute to extensions of the system. Several opportunities for future work will now be discussed.

8.2.1 Enabling the family-based GWAS analysis based on study participants

According to the literature, this is the first time that a pedigree data management has been integrated in a study-based medical data management system. Therefore, transforming the family relationship and family demographic information for the GWAS analysis will facilitate family-based studies in genomic analysis. According to recent literature, there has been a growing demand for conducting family-based GWAS using the extensive pedigree details collected by large biobanking cohorts [230]. There has not been a study-based data management system to

manage and conduct collaborative GWAS on large family datasets which require a sophisticated, systematic approach. The unique features include selecting family members for specific phenotypic attributes, dynamically matching them with existing genomic information, and conducting analysis based on selected pedigree datasets with established analysing techniques.

In addition, dynamically transforming family relationship synonyms based on cultural and language specifications would enable more datasets to be managed in The Ark pedigree module. Implementing interactive pedigree visualisations will provide a more interactive approach to building the pedigree datasets and dynamically extracting information. Developing a web-services interface to import, export and manipulate pedigree information could also be of great benefit, allowing The Ark's pedigree module to interact with other popular pedigree modelling software such as Progeny [40].

8.2.2 Integrating with different genomic datasets and function as heterogeneous genomic data management platform

The Ark phenotype data management module can accommodate personal medical record data. It is essential to store the personal electronic medical records for the personal genomics research space. Development of the common data schema for the data in the electronic medical records and a web interface to interact with datasets visually in the electronic medical records are critical in this phase. Enabling the information in the electronic medical records for the genomic analysis provides a streamlined mechanism to carry out the genomic research on a wide variety of population genetics.

The implementation of analysis algorithms is essential to the genomic research process; analysis cannot proceed unless the algorithm of interest has been programmed. Consequently, developing additional genomic analysis packages for SPARK is a clear line of future work, an open computational package format has been de-

veloped to support this work (see Section 5.4.3). Ultimately, a growing repertoire of analysis packages could be established online, serving as a web-based, open-access repository for researchers worldwide to leverage. The beginnings of this repository already exist at SPHINX web page[1].

The data centre design of the SPARK project enables multiple data sources to be attached to a SPARK node, and therefore a study that has been configured in The Ark. The current implementation of SPARK data centre is capable of managing genomic datasets in a generic sense (GWAS, whole genome sequence, etc.). However, at present, SPARK's columnar genomic data management approach and genomic querying capabilities can only be applied to GWAS datasets in the PLINK format [43]. A natural extension to SPARK's data centre component is to be able to database and process full genome sequence data. This would serve as a particularly useful improvement as the number of full sequence datasets is growing rapidly due to ongoing decreases in sequencing costs [231].

Future work to the SPARK data centre component might also contribute to automated data annotations for genomic datasets and provide a means to extract this meta-data of a dataset based on the annotations. The current solution for GWAS data querying is capable of searching the GWAS data based on individual identity and selected SNP identity. However, it is limited in its ability to carry out additional work on SPARK's data querying capabilities involving specific allele groups with predefined attributes (e.g. predefined allele frequency). This would serve the needs of researchers by approaching multiple aspects of GWAS data management, including data quality control and analysing.

[1] `https://sphinx.org.au/the-ark/spark`

8.2.3 Implementing a SPARK-specific workflow management to automate GWAS analysis

Workflow management is a mechanism that is implemented in popular GWAS systems such as Galaxy. A workflow contains multiple analysis steps which are applied to a genomic dataset, including a preliminary analysis of the dataset, cleaning and quality control, execution of analysis algorithms, and post-processing of results, including packaging the results for retrieval. The current implementation of SPARK is capable of storing workflow steps within a computation package (see Section 5.4.3).

Using this approach, researchers can define a fixed workflow of data processing and analysis steps. However, a formalised and flexible workflow definition mechanism, capable of conditional actions, is not present. SPARK could be improved by the introduction of formal workflow management machismo, capable of complex workflows with many conditional (and potentially parallel) components, and complete with a data model to store the outcomes of individual workflow steps.

To facilitate work on workflow management, an ontology definition that is based on a data model [232] could be used. Using this approach, a web interface could be developed to define workflows in a manner that is re-usable for multiple datasets, and modifiable, serving as the basis for new workflow definitions. The new workflow implementation has the potential to streamline the research process, saving researcher's time by eliminating manual intervention at certain points of the workflow process.

8.2.4 An extensive approach to visualise the GWAS analysis results

Visualisations of genomic data and analysis results can serve as a useful aid to the biomedical research process. The SPARK design has primarily focused on providing a researcher-friendly platform for genomic data management and analysis

enabled by HPC; sophisticated visualisations are outside the scope of this PhD project. As a result, SPARK does not offer the visualisation capabilities found in comparable systems such as Galaxy [45]. To address this limitation inbuilt visualisation techniques could be developed to present the existing datasets residing in the data centres, query output of the selected datasets and the analysis results. This could potentially include visualisations of above-mentioned scenarios using the techniques presented in Integrative Genomic Viewer (IGV) [233] and Circos genome comparison tool [234].

According to the design goals of the IGV platform, genomic data visualisation consists of loading the large datasets, data specific visualisation techniques (GWAS datasets can be visualised as Manhattan plots, and P-value distributions for individual SNPs reside in the dataset), and interactive data presentation. Also, the Circos platform has provided a web-based comparison on genomic dataset and output relationships between the genomic intervals. The large datasets visualisation required significant computation power and mechanisms to integrate with HPC sources to facilitate preprocessing datasets. The SPARK project has a specialised mechanism to integrate with the HPC sources and storing computational packages for analysing. Therefore, integrating external visualisation engines for GWAS data visualisation is a possible enhancement with current SPARK design by storing the visualisation techniques as a repository, connecting multiple visualisation engines based on their availability, and exporting the results based on selected visualisation techniques. Also, there is the possibility to implement the comparison of multiple datasets residing in data centres and visualise their relationships.

8.3 Summary

This thesis has discussed the evolution of the biomedical data management and the next phase of the development goals for this class of systems. Notable advances in genomic research technologies have highlighted the pressing need for advancements

in biomedical tools (see Section 2.4). The rapid evolution of Big Data management and web application architectures (see Section 2.2.3) has provided a technological foundation to meet the challenges of genomic research in the 21st century. The Ark project was chosen as the foundation backbone for the research presented here as it is a prominent open-source biomedical data management system (see Chapter 3). Comparing The Ark with similar biomedical data management systems and examining the requirements of medical research involving genomic data, have helped understand the requirements for SPARK to be developed as a new platform to accommodate studies which can comprise large volumes of non-genomic data (e.g. consent, contact, questionnaire and biospecimen data), pedigree information and multiple genomic datasets. SPARK meets these requirements with a system that manages these data types in an integrated way, i.e. within one system. These requirements also include the need for powerful computing platforms to undertake the different types of complex genomic analyses that are becoming more common in the literature (see Section 2.3.2).

SPARK has introduced a novel application solution for the pedigree (see Chapter 4) and genomic data management that is integrated with The Ark (see Chapter 5), a study-based biomedical data management system. The SPARK design has been supported by new Big Data management techniques, particularly non-relational database models, and microservice architectures applied to web-based systems. The empirical results of Chapter 6 have demonstrated the viability of SPARK's genomic data modelling approach. The SPARK implementation produced by this project benefits the global genomic research community in a highly practical way, promoting collaborative contributions to the software system by way of its open-source licence. Combined, the contributions of Chapters 4, 5, 6, and 7 demonstrate that the SPARK project has satisfied the original project vision (see Section 1.3):

"To enable a novel data management approach for biomedical research datasets, including pedigree and heterogeneous genomic datasets, by enabling massively par-

allel high-performance computing for analysis."

A

Software and Data

Software

The Ark is an Open Source biomedical research data management system developed by collaborating researchers and software developers based in The University of Western Australia and The University of Melbourne. The software project operated under GNU GPL V3 license

The Ark source code available at: `https://github.com/The-Ark-Informatics/ark/`

The Ark documentation for the researchers and software developers are available at: `http://sphinx.org.au/the-ark`

The main Ark modules contributed by this PhD thesis are:

The Ark pedigree module:

A novel study-subject based pedigree data management approach. Available inside The Ark study module (Source: `https://github.com/The-Ark-Informatics/ark/tree/master/ark-study`)

The Ark Genomics module:

A simplified web user interfaces to manage the genomic data and perform complex GWAS analysis inside distributed computing sources. Available in The Ark Genomics module. (Source: `https://github.com/The-Ark-Informatics/ark/tree/master/ark-genomics`)

SPARK:

Distributed computational nodes developed to interface different high-performance computing facilities supporting the native APIs and security constraints. (Source: `https://github.com/The-Ark-Informatics/ark/tree/master/spark-distro`)

Access to the system:

Researchers can access the Live Demo version of The Ark system via the following URL: `https://sphinx.org.au/the-ark/try-it`
The system user manuals to access The Ark-pedigree and genomics modules are available in the following URL: `https://sphinx.org.au/the-ark/documentation`.

Data Sets

Currently, SPARK platform supports only the Plink file formats (BED and PED/MAP), and simulated SNP datasets (10 – 100K) are loaded to The Ark Genomics module.

Prebuilt SPARK specific computational packages available in the following URL: `https://sphinx.org.au/the-ark/spark`

B

SPARK System User Guide

SPARK project is designed as a PhD study aims to build novel software platform to manage modern Genome Wide Association Studies (GWAS). The Ark currently operates as a leading open-source web-based biomedical data management solution for the medical research community (`http://sphinx.org.au/the-ark`). It supports more than 30 medical studies operating locally and internationally managing the phenotypic datasets. The system is capable of handling multiple facets of the medical research (study management, study participant management, questionnaire management, laboratory information management, etc.) data management domain with its web modules.

The Ark users have owned a number of genomic datasets related to existing phenotypic datasets reside there. So there was a discussion to implement a system

based on The Ark. The SPARK project addresses these challenges and is built upon the existing capabilities of The Ark. The Research will examine the properties of Genomic datasets and evaluate a suitable data management platform for GWAS datasets from this experiment. Enabling the high-performance computing is a challenging task for existing data management systems and SPARK project will implement a use case for modern software architecture best practices to integrate with high computational sources (Ex. Supercomputers). As a web-based system, SPARK project will evaluate a GWAS management system for global research community by checking application usability and architecture.

The SPARK project was designed to build upon and extend The Ark framework and its existing data management approach. This was addressed by The Ark Genomics module which operates as a common web interface for a set of micro-services generated by distributed SPARK nodes. Importantly, The Ark is a data management platform rather than analytical pipeline. The SPARK web nodes developed as an independent high-performance Big Data management and high-dimensional analysis extension.

Section A discussed each component of The Ark genomics module and how these components have addressed the objectives of the SPARK project. The Ark Genomics module consists of 4 components to manage existing genomic datasets and conduct an analysis of a selected datasets using pre-configure algorithms. These components are Micro Service, Data Centre, Computation, and Analysis which resides as independent tabs under The Ark Genomics tab.

Section A: Introduction to The Ark Genomics module

Micro Service Tab

The SPARK project has to be compliant with facilitating distributed service location mediator. Therefore, the system needs to record service location with their access credentials. Micro Service tab is designed to declare and manage the iden-

tities of the service endpoints. Here specify the service locations, and researchers can point their analysis to these micro services to execute and observe the experiment results. These micro service entries declare the available services and given opportunity to researchers to select specific services to start an analysis

- Search options available by Micro Service ID, Name, URL, and Status(Online/ Offline).
- Micro Services tab search lists the pre-configured SPARK services with the Id, Name, Description, URL, and Status fields and Test button in the right corner.
- There, click on the Check Status button will update the current status of the service. (Online/ Offline).
- Click on the Name of an individual service reside in the list will navigate to its details.
- Micro Service deletion is omitted in the detail panel, and users can update the description of a service according to their permission level after clicking the Save button.
- Once creating a micro service its URL is disabled to protect the data integrity.
- Click on Cancel button will return to the micro service search list screen.

Data Centre Tab

Genomic datasets are a unique set of data which can be categorised into a type of Big Data. Efficiently manage a genomic dataset, SPARK system consists of lightly coupled service oriented workflows. Data Centres included traversing to distant data sources and examining their content. A data centre section to access and query the data sets attached to data centres. This section is given the opportunity to

researchers to select a micro service and check data centres available with it. Then they can traverse through the data centre file system similar to UNIX operating system by following the root folder structure. The file search is totally based via the Web controllers and eliminates the researcher to familiar with the expensive command to traverse the original file system.

- Select a Micro Service will enable set of Data Centres adhere to its service endpoint.
- After choosing a Micro Service and Data Centre, the researcher can click on search button to navigate to the root directory of the data centre in the top results panel, and data sources related to the selected micro service and data centre in the bottom results panel.
- Data Centre Directory navigation is compliant with Unix file system and start with root (/exampleDir) directory), and click on the name of the directory will traverse to the files and sub-directories of the directory.
- There display the directories and files reside in the directory with File Name, Directory (is a directory or file (yes/no)), Absolute File Name (file path relevance to the root directory), Directory Status (Files ready to process and not required), and Source.
- When navigating to a directory (except the root directory) which contains files with Ready state, researcher can click the Details button against the directory name to create a data source.
- There, researcher can create a data source for the directory/ File which is later referred in the analysis tab.
- After saving a data source, new data source will create in the system, and click the search button again will appear the data source panel in the bottom.
- All the data sources consist with Online, Offline, and Query Buttons.

- Click on the Online button will make the data source to online state which ready for querying.
- The offline button brings the dataset reside in the directory to unprocessed state and enables the online button.
- Click on the Query button opens a dialogue where a researcher can filter the data source based on Id Type (The Ark UID, other linked id type) and id, Family id, Gender, and Affected status.
- Based on the selected output type (PLINK based PED/MAP or Binary format) new physical data directory will be created inside the parent directory and new data source will be added to the data source list.

Computation tab

Sharing different computational analysis is one of the major objectives in SPARK design. The foundation principal of this objective is to develop a single computational package shared and executed on multiple datasets. Operate distinctly; the SPARK system needs to establish a centralised computational package repository. Each package would contain a specific GWAS analysis technique. Analysis technique can be a single program or execution pipeline specific to a computational facility. Computational section manages the computational packages. SPARK design approach provides a common interface to manage different GWAS analysis techniques packaged into the zip archives. Using the computation screen researchers can define a computation package, upload a computation package to an executable environment and optionally compile a package based on the source code

- Search options available by computation ID and Algorithm name and clicking the search button will list down the existing computational packages.
- There display the computational packages with the ID, Algorithm name, Micro Service, and Status (Uploaded/ Processed).

- Against every computational package, there is a Download button to download the source package based on availability.
- Clicking on the Algorithm navigates to existing computational package or click New button allow to create a new computational package.
- Click on Save button allows to create or update computational package details.
- Click on Delete button allows deleting an existing computational package which hasn't already uploaded to a micro service.
- Click on Upload button allows uploading the existing computational package to a micro service.
- The status will change to Uploaded when the package was uploaded and enable the Compile button.
- Click on Compile button Start configuring the computational package in the Micro Service and status updates to Processed when configuration completed (Only use for the source packages).
- Click on the available button will enable the computational package is ready for the analysis.

Analysis Tab

As discussed in Data Centre and Computation sections SPARK provides a simplified dashboard to manage the data sources and computational packages. Combining those two options and execute analysis is the most critical system aspect related to the GWAS. Therefore SPARK consists of an analysis mechanism to execute selected computational packages on a data source. Analysis can be run in a selected service point with compatibility to execute in the environment bound to the computational package. The Meta information set has excluded the data source selection

and needs to be specified per analysis definition via the web interface. The analysis runs as a background job, and status will be updated on event completion. When the analysis is completed, researchers can download the results according to the output file name declared for the analysis. Ultimately researchers would be able to run a single computational package for multiple datasets and compare the results. Analysis section is introduced to execute the computational packages for selected data sources. This process has simplified the traditional long time seeking analysis set-up into a single web page.

- Analysis tab is designed to execute analysis using an available computation package on selected data source in a micro service.
- Search options available by Analysis ID and name and clicking the search button will list down the existing analysis.
- There display the analysis with the ID, name, Micro Service, Data Source, Algorithm, and Status (Submitted/Completed).
- Clicking on the analysis name will navigate to existing analysis or click New button allow to create a new analysis.
- Click on the Save button create an analysis.
- Click on Run Analysis button starts a job and disable immediately after initializing a job.
- When Job completed, enable the Results button to download results file.

Section B: Experiment Steps

Following exercise is designed to get the feedback from prospective researchers, who intend to use The Ark genomics module for GWAS research. To reduce the testing time and validate the results, used the existing 89 HapMap samples and

80K random SNPs dataset provided by the PLINK tutorial (`http://zzz.bwh.harvard.edu/plink/tutorial.shtml`). In addition, data centre operations used the simulated PLINK datasets.

The Ark Genomics module navigation

- Go to the following URL to access The Ark test instance. (`https://demo.sphinx.org.au/ark`)
- Use The Ark username and password to login to the system.
- User Name: sparkuser@ark.org.au Password: Spark@user9
- After successful login to the system, the user will see the list of studies available for login user under the top level Study tab.
- Select the Demo 1 study from the study list by clicking on the study name.

Traverse to the Genomics Module

- Then go to the top level Genomics tab and click on the label.
- There under the Genomics tab shows four sub-tabs called Micro Service, Data Centre, Computation, and Analysis and default select the Micro Service tab.

1.0 Create a new micro service

Step 1.0 will introduce on how to initiate a distributed SPARK node inside The Ark genomics module via a simple web user interface, search the existing micro services, and check their availability.

- Navigate to the Micro Service sub-tab.

- Click on the New Button to create a new micro service with following credentials.
- Name: Micro-Service-XX (Should be a unique name)
- URL: `http://115.146.84.128:8080/spark-demo`
- Description: Custom description of your service
- Click on Save button to create a new service endpoint.
- Click Cancel button to return to the micro service list which displays the newly created micro service.
- Check the status of the service by clicking the Check status button (Service available = Status (Online)).

2.0 Navigate, create and query data sources

Step 2 and sub-steps will guide the user on how to navigate to specific file or directory inside a data centre which is similar to Unix directory structure with web-based navigation. Then create a data source which will be later used to query by parameters (id, gender, affected status, etc) and analysis (Step 4)

- Click on the Data Centre tab
- Select a previously created Micro Service from drop down list (identify a micro service by the name).
- Select the Data Center from the drop down list (Which would be only enabled after selecting a micro service).
- Click on Search button to navigate to the root directory of the Data Centre.

- In the top search results shows plain, binary, and hapmap1 directories. Each directory consists of the PLINK format GWAS datasets. In the Plain and binary directories consist of sub directories structures with SNP count and population. Hapmap1 directory consist of sample GWAS dataset which will be future used for the analysis.

2.1 Navigate to Directory or File

- Navigate to a directory, Click on the File Name of the directory to see the files and sub-directories. Use the Back button in the top left hand corner of the grid to traverse back to parent directory.

2.2 Create data source

- Traverse to the Plain - > 100SNP by clicking the file name. Click detail button of the 1000 directory and will pop-up the Details pop-up window.
- Fill the Meta information fields (Description, Type, Owner, Chip etc) with available information, and non-of the fields are mandatory. Then, click Save button to create a new data source. (If data source is already created for the directory/file will indicate by displaying the ID value)
- Also, follow the same set of steps mentioned above to create a data source for hapmap1 directory, which will be later used in the analysis experiment.
- Click on the search button, then new data source record is visible in the bottom results panel.

2.3 Query data source

- To query a data source, click on online button to process the data source and make data source ready for querying.

- When data source is ready to query, enables the query button, researcher can click the query button to open the dialog to set query parameters.
- There, researcher can set query the dataset by The Ark Subject UID, Other Id, family id, gender, and affected status. In addition, query option enabled to set the PLINK based output type (PED/MAP or BIN) for the result data source.
- When the query operation has completed, there is a new data source would be added to the data source list and sub- directory would be created inside the data centre with the source id in the original data source directory. Note, user need to click the search button to refresh the data source results list panel to see the newly created data source.

3.0 Upload Computation package

Step 3 will introduce user to create a computation package which include analysis instructions as a source code, bash script or single command, and SPARK.info file with the execution Meta information. Then computation package can be uploaded to the selected micro service and make available for analysis.

- Click on the Computation tab
- Download existing Basic_Association_Analysis package by clicking on Download button from the computation package list to your preferred download directory.
- Click New button to create a new algorithm package for analysis.
- Fill required fields including Name Text field and select previously created (Refer to step 1.0) Micro Service from drop down list.
- Attach the previously downloaded SNP- algorithm source archive as the Algorithm package by clicking on the Browse button.

- Click Save button to save the Computational package and Click Cancel button to return to Computation list.
- Click on Upload button to upload the package to micro service.
- When Computational status has changed to Uploaded then click Available button on Algorithm name to enable computation package for analysis.

4.0 Run Analysis and download results

Step 4 make researchers to select a micro service, data source, and computation package along with specifying the micro service specific analysis parameters and analysis output. When analysis is executed user can monitor the progress by refreshing the screen and download the results of completed analysis.

- Click on the Analysis tab
- Click New button to create a new analysis.
- Fill required fields by the following order. Assign analysis name by filling Name Text field, selecting your service from Micro Service drop-down list, previously created computation package from Algorithm drop-down list, selecting "/hapmap1" data source from Data Source drop-down list (Refer to step 2.2), and specifying the analysis output in Result File Name text field as 'hap.assoc', and Parameters copied from existing Hapmap_Basic_Assoc analysis configured in spark-demo micro service.
- Click Save button to initiate the analysis.
- Click Cancel button to return to the Analysis search result list.
- Click Run Analysis button to execute an analysis then automatically update the analysis status and job id return from the Job Queue.

- Click the Search button to refresh the analysis results list
- When the analysis status has updated to ”Completed” then click Download Results button to download the analysis output.
- Output can be compared with the results explained in the PLINK Tutorial mentioned above.

Special Notes

- The Ark demo instance resets at 00:00 AEDT every day. Therefore, saved experiment results, configurations, and intermediary steps saved only one day inside the system.
- If you get any questions Or Issues following the experiment steps please contact kranaweera@student.unimelb.edu.au

C

SPARK User Acceptance Testing

Instructions

Please complete the following questions. The information that you provide will be kept strictly confidential.
The following point scale reflects your response to each question:

1. Strongly Disagree
2. Disagree
3. Undecided/Neutral
4. Agree

5. Strongly Agree

PART A: General Usability

1. I think that I would like to use this system frequently.

    1. Strongly Disagree
    2. Disagree
    3. Undecided/Neutral
    4. Agree
    5. Strongly Agree

2. I found the system unnecessarily complex.

    1. Strongly Disagree
    2. Disagree
    3. Undecided/Neutral
    4. Agree
    5. Strongly Agree

3. I thought the system was easy to use.

    1. Strongly Disagree
    2. Disagree
    3. Undecided/Neutral
    4. Agree

5. Strongly Agree

4. I think that I would need the support of a technical person to be able to use this system.

   1. Strongly Disagree
   2. Disagree
   3. Undecided/Neutral
   4. Agree
   5. Strongly Agree

5. I found the various functions in this system were well integrated.

   1. Strongly Disagree
   2. Disagree
   3. Undecided/Neutral
   4. Agree
   5. Strongly Agree

6. I thought there was too much inconsistency in this system.

   1. Strongly Disagree
   2. Disagree
   3. Undecided/Neutral
   4. Agree
   5. Strongly Agree

7. I would imagine that most people would learn to use this system very quickly.
    1. Strongly Disagree
    2. Disagree
    3. Undecided/Neutral
    4. Agree
    5. Strongly Agree
8. I found the system very cumbersome to use.
    1. Strongly Disagree
    2. Disagree
    3. Undecided/Neutral
    4. Agree
    5. Strongly Agree
9. I felt very confident using the system.
    1. Strongly Disagree
    2. Disagree
    3. Undecided/Neutral
    4. Agree
    5. Strongly Agree
10. I needed to learn a lot of things before I could get going with this system.
    1. Strongly Disagree

    2. Disagree
    3. Undecided/Neutral
    4. Agree
    5. Strongly Agree

PART B: SPARK-specific Usability

1. It is easy to configure a new micro service.
    1. Strongly Disagree
    2. Disagree
    3. Undecided/Neutral
    4. Agree
    5. Strongly Agree
2. The status of a micro service is clearly indicated and understood.
    1. Strongly Disagree
    2. Disagree
    3. Undecided/Neutral
    4. Agree
    5. Strongly Agree
3. It is easy to navigate to a specific data source.
    1. Strongly Disagree

2. Disagree
3. Undecided/Neutral
4. Agree
5. Strongly Agree

4. I understand the meaning of the online/offline status for a dataset.

1. Strongly Disagree
2. Disagree
3. Undecided/Neutral
4. Agree
5. Strongly Agree

5. I understand how to record meta-information for Genome Wide Association Study (GWAS) data.

1. Strongly Disagree
2. Disagree
3. Undecided/Neutral
4. Agree
5. Strongly Agree

6. I am confident in extracting genotype records using the query builder.

1. Strongly Disagree
2. Disagree

3. Undecided/Neutral

4. Agree

5. Strongly Agree

7. I understand how to install a new computational package.

1. Strongly Disagree

2. Disagree

3. Undecided/Neutral

4. Agree

5. Strongly Agree

8. I understand the meaning of the uploaded/installed status of a computational package.

1. Strongly Disagree

2. Disagree

3. Undecided/Neutral

4. Agree

5. Strongly Agree

9. I am confident in performing analyses using the available computational packages and data sources.

1. Strongly Disagree

2. Disagree

3. Undecided/Neutral
4. Agree
5. Strongly Agree

10. I am able to extract the results of an analysis.

1. Strongly Disagree
2. Disagree
3. Undecided/Neutral
4. Agree
5. Strongly Agree

PART C: General Feedback

Comments

D

SPARK Functional Test Cases

Test Run Purpose:

Approve The Ark Genomics module micro service functionality

Specification Reference:

Micro service usability guide available in The Ark specifications guides.

Test Case Preparation:

Check The Ark system is available and access by spark user credentials.

Description:

Need to evaluate the The Ark genomics module micro service function capabilities with creating , updating, searching, and checking the availability of the micro services deployed via the distributed SPARK modules.

1.0 Micro Service Test Cases:

ID:

1.1

Test Details:

Check The valid Ark user can access the micro service tab

Input Data:

Login: sparkuser@ark.org.au

Result:

Pass

Comments or Test Issue Description:

Login to The Ark system using demo.sphinx.org.au/ark with the following user name: sparkuser@ark.org.au. Select Demo1 study by clicking the Study name. Then sparkuser@ark.org.au user should be able to traverse to Genomics -> Micro Service tab.

ID:

1.2

Test Details:

Search Micro Service without search parameters.

Input Data:

Result:

Pass

Comments or Test Issue Description:

List all the micro service entries saved in The Ark system. There is only 2 micro services are configured in the demo system and should be visible in the search list panel after click the search button.

ID:

1.3

Test Details:

Search Micro Service with available id.

Input Data:

ID:2

Result:

Pass

Comments or Test Issue Description:

Enter the existing micro service entity id (2), and click the search button. List only the micro service entity contains the id 2.

ID:

1.4

Test Details:

Search Micro Service with id not-existing in the system.

Input Data:

ID:15

Result:

Pass

Comments or Test Issue Description:

Enter the non-existing micro service entity id (15), and click the search button. Search result shows an empty search panel.

ID:

1.5

Test Details:

Search Micro Service with full name existing in the system.

Input Data:

Name:spark-demo

Result:

Pass

Comments or Test Issue Description:

Enter the existing micro service entity name (spark-demo), and click the search button. List only the micro service entity contains the name spark-demo.

ID:

1.6

Test Details:

Search Micro Service with partial name existing in the system.

Input Data:

Name: demo

Result:

Pass

Comments or Test Issue Description:

Enter the existing micro service entity name partially (demo), and click the search button. List only the micro service entity contains the name demo (spark-demo).

ID:

1.7

Test Details:

Search Micro Service with name not-existing in the system.

Input Data:

Name: Super

Result:

Pass

Comments or Test Issue Description:

Enter the non-existing micro service entity name (super), and click the search button. Search result shows an empty search panel.

ID:

1.8

Test Details:

Search Micro Service with URL existing in the system.

Input Data:

URL: http://localhost:8080/spark-local

Result:

Pass

Comments or Test Issue Description:

Enter the existing micro service URL (http://localhost:8080/spark-local), and click the search button. List only the micro service entity contains the URL (spark-local).

ID:

1.9

Test Details:

Search Micro Service with partial URL existing in the system.

Input Data:

URL: 115.146

Result:

Pass

Comments or Test Issue Description:

Enter the partial micro service URL (115.146) exists in the system, and click the search button. List only the micro service entity contains the URL (spark-demo).

ID:

1.10

Test Details:

Search Micro Service with URL not exists in the system

Input Data:

URL: http://155.16.14.55/spark-sys

Result:

Pass

Comments or Test Issue Description:

Enter the non existing micro service URL (http://155.16.14.55/spark-sys), and click the search button. . Search result shows an empty search panel.

ID:

1.11

Test Details:

Search Micro Service with status Online

Input Data:

Status: Online

Result:

Pass

Comments or Test Issue Description:

Choose Onine option in the Status drop down list), and click the search button. . List only the micro service status is Online (spark-local).

ID:

1.12

Test Details:

Search Micro Service with status Offline

Input Data:

Status: Offline

Result:

Pass

Comments or Test Issue Description:

Choose Offine option in the Status drop down list), and click the search button. . List only the micro service status is Offline (spark-demo).

ID:

1.13

Test Details:

Click the Check Status button to check the recent status of micro service

Input Data:

Activate the spark-demo service

Result:

Pass

Comments or Test Issue Description:

Click on the Check Status button assigned to spark-demo Micro service after activating the service. Spark-demo micro service status changed to Offline to Online.

ID:

1.14

Test Details:

Create new micro service with unique name and required fields.

Input Data:

Name=Spark-test, URL=http://115.146.84.128:8080/spark-demo, Description=Test micro service created

Result:

Pass

Comments or Test Issue Description:

Click on the New button to create a new Micro service and enter the details panel. There fill the input fields with the specified input in following order (Name, URL, and Description). Then click Save button. On top of the details panel shows the 'Micro service has created successfully'. Click cancel will return to the search panel, and there display newly created spark-test micro service with status is online.

ID:

1.15

Test Details:

Not Allowing editing the already created micro service name and URL

Input Data:

Result:

Pass

Comments or Test Issue Description:

Click on spark-test micro service in the list panel and enter the details screen. There, disble the Name and URL fields of the micro service. Can edit only the Description field.

ID:

1.16

Test Details:

Not Allowing creating new micro service with name already exists

Input Data:

Name=Spark-test, URL=http://115.146.84.128:8080/spark-demo, Description=Test micro service created

Result:

Pass

Comments or Test Issue Description:

Click on the New button to create a new Micro service and enter the details panel. There fill the input fields with the specified input in following order (Name, URL, and Description). Then click Save button. Warning message display the "Micro service name is already exists".

ID:

1.17

Test Details:

Not Allowing creating a micro service without name and URL

Input Data:

Description=Test micro service description

Result:

Pass

Comments or Test Issue Description:

Click on the New button to create a new Micro service and enter the details panel. Enter only the description and click the save button. Warning message is display in the top of the panel with “Required Name”, and “Required URL”.

Test Run Purpose:

Approve The Ark Genomics module Data Centre functionality

Specification Reference:

Data Centre usability guide available in The Ark specifications guides.

Test Case Preparation:

Check The Ark system is available and access by spark user credentials. Need to place PLINK formatted GWAS datasets inside the data centre root directory structure.

Description:

Need to evaluate the The Ark genomics module data centre function capabilities with searching the available datasets reside in the data centres attached to micro services, create, edit, and delete the data sources based on existing data set Meta information, and searching the GWAS related information in the data sources.

2.0 Data Centre Test Cases:

ID:

2.1

Test Details:

Check The valid Ark user can access the Data Centre tab

Input Data:

Login as sparkuser@ark.org.au

Result:

Pass

Comments or Test Issue Description:

Login to The Ark system using demo.sphinx.org.au/ark with the following user name: sparkuser@ark.org.au. Select Demo1 study by clicking the Study name. Then sparkuser@ark.org.au user should be able to traverse to Genomics -> Data Centre tab.

ID:

2.2

Test Details:

Search For a data centre available in micro service

Input Data:

Micro Service= SPARK-Demo

Result:

Pass

Comments or Test Issue Description:

Select the SPARK-Demo micro service from the back end. In the Data Centre drop down list is populate with the SPARK-SSH data centre.

ID:

2.3

Test Details:

Search For a data centre in a disabled micro service

Input Data:

Micro Service= SPARK-Demo, Disable the SPARK-Demo micro service

Result:

Pass

Comments or Test Issue Description:

Select the SPARK-Demo micro service from the back end. In the Data Centre drop down list should be empty.

ID:

2.4

Test Details:

Navigate to a specific directory inside a data centre .

Input Data:

Micro Service= SPARK-Demo, Data Centre=SPARK-SSH

Result:

Pass

Comments or Test Issue Description:

In the list of directories inside the previously selected data centre (2.3) , click on 100 SNP File name in the list. Then system will navigate to the sub directories and files inside the 100SNP directory.

ID:

2.5

Test Details:

Check the back button is disabled when navigate back to data centre root directory

Input Data:

Micro Service= SPARK-Demo, Data Centre=SPARK-SSH

Result:

Pass

Comments or Test Issue Description:

Select the micro service and data centre, then click search button. Back button should be disabled. When navigate to directory (1.1.4), then Back button should be enabled. Navigate to the last directory of the sub directory structure. When click the back button always navigate back to previous directory, and disable in the root directory.

ID:

2.6

Test Details:

Create a data source

Input Data:

Micro Service= SPARK-Demo, Data Centre=SPARK-SSH, Description =Sample dataset, type = PLINK, Owner= Admin, Chip= Illumina, SNP count= 100, size = 12 MB

Result:

Pass

Comments or Test Issue Description:

Select the input micro service and data centre, and click on search button. In the directory list displayed in top search h panel, click on 100SNP Directory name and traverse into the sub-directories. There. Click the the Details button in the 100 directory which contains the 100 SNP, 100 Population dataset. In the Details, Pop-up dialog provide input to Description, data set type, Owner, Chip, SNP count, and size of the dataset ("Sample dataset", Plink, "Admin", "Illumina", 100, 12 MB). Then, click save button and dialog will close. When click the search button newly created data source is display in the bottom results panel with the name of the physical directory.

ID:

2.7

Test Details:

Edit existing data source

Input Data:

Micro Service= SPARK-Demo, Data Centre=SPARK-SSH,Owner= Demo

Result:

Pass

Comments or Test Issue Description:

Repeat the steps in 1.1.6 to traverse to the directory and click on Details button. In the Details pop-up dialog would display the values entered in previous step.

Change the Owner to Demo and click the save button. Then click again the Details button, will show following details matching with input to Description, data set type, Owner, Chip, SNP count, and size of the dataset (“Sample dataset”, Plink, “Demo”, “Illumina”, 100, 12 MB)

ID:

2.8

Test Details:

Delete existing data source

Input Data:

Micro Service= SPARK-Demo, Data Centre=SPARK-SSH

Result:

Pass

Comments or Test Issue Description:

Repeat the steps in 1.1.6 to traverse to the directory and click on Details button. In the Details pop-up dialog would display the values entered in previous step. Click on the delete button then Confirm Are you sure you want to delete, Then dialog will close, Search the data source id, which cannot be display in the data source search panel.

ID:

2.9

Test Details:

Make data source online

Input Data:

Micro Service= SPARK-Demo, Data Centre=SPARK-SSH

Result:

Pass

Comments or Test Issue Description:

Click the search button and select a datasource in the lower search panel. Click on the Online button and then it will disable the online button, keep Offline, Query buttons in disable mode. Periodically refresh the result panels by clicking the search button. When online process has completed, disable the Online button and enable the Offline, and Query buttons.

ID:

2.10

Test Details:

Make online data source offline

Input Data:

Micro Service= SPARK-Demo, Data Centre=SPARK-SSH

Result:

Pass

Comments or Test Issue Description:

Click the Offline button of the already online data source. Which disable the Online and Query buttons. Periodically refresh the result panels by clicking the search button. When Offline process has completed, enable the Online button and disable the Offline, and Query buttons.

ID:

2.11

Test Details:

Query data source for existing Individual UID

Input Data:

Micro Service= SPARK-Demo, Data Centre=SPARK-SSH, Id Type = UID, Individual Id=Per0, Output type = PED/MAP

Result:

Pass

Comments or Test Issue Description:

Click the Query button of the already online data source. This enable the pop-up Query dialog box. Select the Id type as the UID and Enter Per0 as the Individual

Id. Select the Output type as PED/MAP, and click query button. Pop-up dialoag will close. Refresh the search results by clicking the search button will enable new data source without enabling the online/offline/query buttons. When query process has completed , then enable the Online button for the data source. Also new sub-directory has created inside (100SNP/10 directory) with data source id in the parent directory with the name combination of Query<DataSourceId>. Inside PED file with only Individual Id per0 data and MAP file with all the SNP information.

ID:

2.12

Test Details:

Query data source for Other ID Type

Input Data:

Micro Service= SPARK-Demo, Data Centre=SPARK-SSH, Id Type = GWAS1, , Output type = PED/MAP

Result:

Pass

Comments or Test Issue Description:

In subject -> Demographic Data details panel define other id for couple of subjects with source as GWAS1 and ids as (per0,per1, per2). Click the Query button of the already online data source. This enable the pop-up Query dialog box. Select the Id type as the GWAS1. Select the Output type as PED/MAP, and click

query button. Pop-up dialoag will close. Refresh the search reults by clicking the search button will enable new data source without enabling the online/offline/query buttons. When query process has completed , then enable the Online button for the data source. Also new sub-directory has created inside (100SNP/10 directory) with data source id in the parent directory with the name combination of Query<DataSourceId>. >. Inside PED file with per0,per1, and2 data which belong to GWAS1 other id and MAP file with all the SNP information.

ID:

2.13

Test Details:

Query data source for family id

Input Data:

Micro Service= SPARK-Demo, Data Centre=SPARK-SSH, family Id = Per0, , Output type = PED/MAP

Result:

Pass

Comments or Test Issue Description:

In subect -> Demographic Data details panel define family id for a subect with per0. Click the Query button of the already online datasource. This enable the pop-up Query dialog box. Input Per0 as the family Id. Select the Output type as PED/MAP, and click query button. Pop-up dialoag will close. Refresh the search

results by clicking the search button will enable new data source without enabling the online/offline/query buttons. When query process has completed , then enable the Online button for the data source. Also new sub-directory has created inside (100SNP/10 directory) with data source id in the parent directory with the name combination of Query<DataSourceId>. >. Inside PED file with only per0 data and MAP file with all the SNP information.

ID:

2.14

Test Details:

Query data source for Gender Type

Input Data:

Micro Service= SPARK-Demo, Data Centre=SPARK-SSH, Gender Type = Male, , Output type = PED/MAP

Result:

Pass

Comments or Test Issue Description:

Click the Query button of the already online data source. This enable the pop-up Query dialog box. Select gender type as Male. Select the Output type as PED/MAP, and click query button. Pop-up dialoag will close. Refresh the search results by clicking the search button will enable new data source without enabling the online/offline/query buttons. When query process has completed , then enable

the Online button for the data source. Also new sub-directory has created inside (100SNP/10 directory) with data source id in the parent directory with the name combination of Query<DataSourceId>. >. Inside PED file with individuals gender type is Male and MAP file with all the SNP information.

ID:

2.15

Test Details:

Query data source for Affected Status

Input Data:

Micro Service= SPARK-Demo, Data Centre=SPARK-SSH, Affected status = Case, , Output type = PED/MAP

Result:

Pass

Comments or Test Issue Description:

Click the Query button of the already online data source. This enable the pop-up Query dialog box. Select Affected status as Case. Select the Output type as PED/MAP, and click query button. Pop-up dialoag will close. Refresh the search results by clicking the search button will enable new data source without enabling the online/offline/query buttons. When query process has completed , then enable the Online button for the data source. Also new sub-directory has created inside (100SNP/10 directory) with data source id in the parent directory

with the name combination of Query<DataSourceId>. >. Inside PED file with individuals affected status type is Case and MAP file with all the SNP information.

ID:

2.16

Test Details:

Query data source for invalid Individual UID

Input Data:

Micro Service= SPARK-Demo, Data Centre=SPARK-SSH, Id Type = UID, Individual Id=Per11, , Output type = PED/MAP

Result:

Pass

Comments or Test Issue Description:

Click the Query button of the already online datasource. This enable the pop-up Query dialog box. Select the Id type as the UID and Enter Per111 as the Individual Id. Select the Output type as PED/MAP, and click query button. Pop-up dialoag will close. Refresh the search results by clicking the search button will enable new data source without enabling the online/offline/query buttons. When query process has completed , then enable the Online button for the data source. Also new sub-directory has created inside (100SNP/10 directory) with data source id in the parent directory with the name combination of Query<DataSourceId>. Inside only the MAP file with all the SNP information is created.

Test Run Purpose:

Approve The Ark Genomics module Computation functionality

Specification Reference:

Computation usability guide available in The Ark specifications guides.

Test Case Preparation:

Check The Ark system is available and access by spark user credentials.

Description:

Need to evaluate the The Ark genomics module computation function capabilities with creating , updating, searching, and checking the availability of the computation deployed in selected micro service.

3.0 Computation Test Cases:

ID:

3.1

Test Details:

Check The valid Ark user can access the computation tab

Input Data:

Login as sparkuser@ark.org.au

Result:

Pass

Comments or Test Issue Description:

Login to The Ark system using demo.sphinx.org.au/ark with the following user name: sparkuser@ark.org.au. Select Demo1 study by clicking the Study name. Then sparkuser@ark.org.au user should be able to traverse to Genomics -> Computation tab.

ID:

3.2

Test Details:

Search computation packages without search parameters.

Input Data:

Result:

Pass

Comments or Test Issue Description:

List all the computation entries saved in The Ark system. There is only 1 computation package is configured in the demo system and should be visible in the search list panel after click the search button.

ID:

3.3

Test Details:

Search computation package with available id.

Input Data:

Id= 2

Result:

Pass

Comments or Test Issue Description:

Enter the existing computation entity id (2), and click the search button. List only the computation entity contains the id 2.

ID:

3.4

Test Details:

Search computation package with id not-existing in the system.

Input Data:

Id= 15

Result:

Pass

Comments or Test Issue Description:

Enter the non-existing computation entity id (15), and click the search button. Search result shows an empty search panel.

ID:

3.5

Test Details:

Search computation package with full name existing in the system

Input Data:

Algorithm Name= Basic_Association_Analysis

Result:

Pass

Comments or Test Issue Description:

Enter the existing computation Algorithm name (Basic_Association_Analysis), and click the search button. List only the computation entity contains the algorithm name.

ID:

3.6

Test Details:

Search computation with partial name existing in the system.

Input Data:

Algorithm Name= Basic

Result:

Pass

Comments or Test Issue Description:

Enter the existing computation entity name partially (Basic), and click the search button. List only the micro service entity contains the name Basic (Basic_Association_Analysis).

ID:

3.7

Test Details:

Search with computation name not-existing in the system.

Input Data:

Algorithm Name= Super

Result:

Pass

Comments or Test Issue Description:

Enter the non-existing entity name (super), and click the search button. Search result shows an empty search panel.

ID:

3.8

Test Details:

Create Computation with Algorithm name, micro service, and algorithm package

Input Data:

Algorithm Name=Test_analysis, Description=Test computation description, Micro Service = demo-service, package= assoc_analysis.zip

Result:

Pass

Comments or Test Issue Description:

Click the New button and enter the computation detail panel. There, enter the algorithm name and description. Then, select a available micro service and algorithms package stored in the local computer. Click save button, then in the search results panel show a new computation entity with new id.

ID:

3.9

Test Details:

Edit Computation with Algorithm name, micro service, and algorithm package

Input Data:

Algorithm Name=Test_analysis_change

Result:

Pass

Comments or Test Issue Description:

Click on existing algorithm name Test_analysis , then enter the details screen of an existing computation package. Change the Algorithm name to Test_analysis_-change. After that click save button and return to the search result list panel. There, computation package will be listed with new name.

ID:

3.10

Test Details:

Delete existing Computation package

Input Data:

Algorithm Name=Test_analysis_change

Result:

Pass

Comments or Test Issue Description:

Click on existing algorithm name Test_analysis , then enter the details screen of an existing computation package. Click delete button and confirm the delete operation by clicking yes in the confirmation dialog. Return to the search panel and could not find the existing computation package in the list.

ID:

3.11

Test Details:

Upload already created computation package to available Micro Service

Input Data:

Algorithm Name=Test_analysis

Result:

Pass

Comments or Test Issue Description:

Search for a computation package which upload button is enabled. Click on the upload button, then disable the upload button and status changed to Processing. Refresh the computation list, when uploading finished change the status to Uploaded and keep disable the upload button.

ID:

3.12

Test Details:

Make available already created computation package

Input Data:

Algorithm Name=Test_analysis

Result:

Pass

Comments or Test Issue Description:

Click on the Available button, then change the button label to unavailable. Go to Analysis tab and click new button and select the package based micro-service, then drop down list should consist the package.

ID:

3.13

Test Details:

Try to upload already created computation package to offline Micro Service

Input Data:

Disable demo-service

Result:

Pass

Comments or Test Issue Description:

Click on the upload button, then disable the upload button and status changed to Failed.

ID:

3.14

Test Details:

Make unavailable an available computation package

Input Data:

Algorithm Name=Test_analysis

Result:

Pass

Comments or Test Issue Description:

Click on the Unavailable button, then change the button label to available. Go to Analysis tab and click new button and try to select the package based micro-service, then drop down list should not show the package.

ID:

3.15

Test Details:

Try to delete an already uploaded computation package

Input Data:

Algorithm Name=Test_analysis

Result:

Pass

Comments or Test Issue Description:

Click on the computation package name which status is Uploaded. There, message will display that the uploaded computation package cannot be deleted.

Test Run Purpose:

Approve The Ark Genomics module Analysis functionality

Specification Reference:

Analysis usability guide available in The Ark specifications guides.

Test Case Preparation:

Check The Ark system is available and access by spark user credentials.

Description:

Need to evaluate the The Ark genomics module analysis function capabilities with creating , updating, searching, executing, and checking the status of the analysis run in selected micro service.

4.0 Analysis Test Cases:

ID:

4.1

Test Details:

Check The valid Ark user can access the analysis tab

Input Data:

Login as sparkuser@ark.org.au

Result:

Pass

Comments or Test Issue Description:

Login to The Ark system using demo.sphinx.org.au/ark with the following user name: sparkuser@ark.org.au. Select Demo1 study by clicking the Study name. Then sparkuser@ark.org.au user should be able to traverse to Genomics -> Analysis tab.

ID:

4.2

Test Details:

Search analysis without search parameters.

Input Data:

Result:

Pass

Comments or Test Issue Description:

List all the analysis entries saved in The Ark system. There is only 1 analysis is executed in the demo system and should be visible in the search list panel after click the search button.

ID:

4.3

Test Details:

Search analysis with available id.

Input Data:

Id=2

Result:

Pass

Comments or Test Issue Description:

Enter the existing analysis entity id (2), and click the search button. List only the analysis entity contains the id 2.

ID:

4.4

Test Details:

Search analysis with id not-existing in the system.

Input Data:

Id=18

Result:

Pass

Comments or Test Issue Description:

Enter the non-existing analysis entity id (18), and click the search button. Search result shows an empty search panel.

ID:

4.5

Test Details:

Search analysis full name existing in the system.

Input Data:

Name=Hapmap_Basic_Assoc

Result:

Pass

Comments or Test Issue Description:

Enter the existing analysis entity name (Hapmap_Basic_Assoc), and click the search button. List only the analysis entity contains the name.

ID:

4.6

Test Details:

Search analysis with partial name existing in the system.

Input Data:

Name=Basic

Result:

Pass

Comments or Test Issue Description:

Enter the existing analysis entity name partially (Basic), and click the search button. List only the analysis entity contains the name Basic (Hapmap_Basic_Assoc).

ID:

4.7

Test Details:

Search with analysis name not-existing in the system.

Input Data:

Name=super

Result:

Pass

Comments or Test Issue Description:

Enter the non-existing analysis entity name (super), and click the search button. Search result shows an empty search panel.

ID:

4.8

Test Details:

Create analysis with name, micro service, Data source, Output File name, parameters, and algorithm package.

Input Data:

Name=Test_analysis, Description=Test analysis description, Micro Service = demo-service, data source =hapmap1, Algorithm= Basic_Association_analysis, Result File Name = hap.assoc, parameters = –file hapmap1/hapmap1 –assoc –noweb –out

Result:

Pass

Comments or Test Issue Description:

Click the New button and enter the analysis detail panel. There, enter the analysis name and description. Then, select a avaialable micro service, computation package, data source from drop down lists. Then insert the parameters required for analysis and output file name . Click save button, thenin the search results panel show a new analysis entity with new id.

ID:

4.9

Test Details:

Edit analysis with status is Undefined

Input Data:

Name=Test_analysis_change

Result:

Pass

Comments or Test Issue Description:

Click on existing analysis name Test_analysis , then enter the details screen of an existing analysis. Change the analysis name to Test_analysis_change. After that click save button and return to the search result list panel. There, analysis will be listed with new name.

ID:

4.10

Test Details:

Delete analysis with status is Undefined

Input Data:

Result:

Pass

Comments or Test Issue Description:

Click on existing analysis name Test_analysis , then enter the details screen of an existing analysis. Click delete button and confirm the delete operation by clicking yes in the confirmation dialog. Return to the search panel and could not find the existing computation package in the list.

ID:

4.11

Test Details:

Run analysis with status is Undefined

Input Data:

Result:

Pass

Comments or Test Issue Description:

Search for an analysis which is already created in the search list. Click on the Run Analysis button, then disable the Run Analysis button and status changed to Processing. Refresh the analysis list, when analysis execution finished change the status to completed and keep disable the Run Analysis button. Then, enable the Download Results button.

ID:

4.12

Test Details:

Download Completed analysis

Input Data:

Result:

Pass

Comments or Test Issue Description:

Click on the Download Results button, on a analysis which status is completed . Results file should be downloaded with the given output name in the analysis.

ID:

4.13

Test Details:

Try to download non-completed analysis results

Input Data:

Result:

Pass

Comments or Test Issue Description:

Search for a analysis which status is not completed. Try to click on the Download Results button. Always, Download Results button should be disabled for the non-

completed analysis.

Bibliography

[1] Ikram MK, Xueling S, Jensen RA, Cotch MF, Hewitt AW, Ikram MA, et al. Four Novel Loci (19q13, 6q24, 12q24, and 5q14) Influence the Microcirculation In Vivo. PLoS Genet. 2010 10;6(10).

[2] SeqWare User Tutorial;. Accessed: 2017-08-16. https://seqware.github.io/docs/3-getting-started/user-tutorial/.

[3] PLINK File Format Reference;. Accessed: 2017-07-12. https://www.cog-genomics.org/plink2/formats.

[4] PLINK Flat files (MAP/PED);. Accessed: 2017-07-15. http://www.gwaspi.org/?page_id=145.

[5] Running the Pipeline Initial Process);. Accessed: 2017-05-14. https://rdp.cme.msu.edu/tutorials/init_process/RDPtutorial_INITIAL-PROCESS.html.

[6] Guimaraes V, Hondo F, Almeida R, Vera H, Holanda M, Araujo A, et al. A study of genomic data provenance in NoSQL document-oriented database systems. In: Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on. IEEE; 2015. p. 1525–1531.

[7] Aniceto R, Xavier R, Guimarães V, Hondo F, Holanda M, Walter ME, et al. Evaluating the Cassandra NoSQL database approach for genomic data persistency. International journal of genomics. 2015;2015.

[8] Carpenter KJ. The history of scurvy and vitamin C. Cambridge University Press; 1988.

[9] Johnson S. The ghost map: The story of London's most terrifying epidemic–and how it changed science, cities, and the modern world. Penguin; 2006.

[10] Watson JD, Crick FH, et al. Molecular structure of nucleic acids. Nature. 1953;171(4356):737–738.

[11] Prinz GA. Magnetoelectronics. Science. 1998;282(5394):1660–1663.

[12] Lodish H, Berk A, Zipursky SL, Matsudaira P, Baltimore D, Darnell J, et al. Molecular cell biology. vol. 3. Scientific American Books New York; 1995.

[13] Moore GE. Cramming More Components Onto Integrated Circuits. Proceedings of the IEEE. 1998;86(1):82–85.

[14] Amdahl GM. Validity of the single processor approach to achieving large scale computing capabilities. In: Proceedings of the April 18-20, 1967, spring joint computer conference. ACM; 1967. p. 483–485.

[15] Codd EF. A Relational Modle of Data for Large Shared Data Banks. Communications of the ACM. 1970;13(6):377–387.

[16] Kernighan BW, Ritchie DM. The C programming language; 2006.

[17] Stroustrup B. The C++ programming language. Pearson Education India; 1995.

[18] Arnold K, Gosling J, Holmes D. The Java programming language. Addison Wesley Professional; 2005.

[19] Scollo C, Shumann S. Professional PHP programming. Wrox Press Ltd.; 1999.

[20] Bächle M, Kirchberg P. Ruby on rails. IEEE software. 2007;24(6).

[21] Berman JJ. Biomedical Informatics. Jones & Bartlett Learning, LLC; 2007.

[22] McKusick VA, Ruddle FH. A new discipline, a new name, a new journal. Genomics. 1987;1(1):1–2.

[23] Burton PR, Tobin MD, Hopper JL. Key concepts in genetic epidemiology. The Lancet. 2005;366(9489):941–951.

[24] Hall JM, Lee MK, Newman B, Morrow JE, Anderson LA, Huey B, et al. Linkage of early-onset familial breast cancer to chromosome 17q21. Science. 1990;250(4988):1684–1689.

[25] Collins FS, Morgan M, Patrinos A. The Human Genome Project: lessons from large-scale biology. Science. 2003;300(5617):286–290.

[26] Consortium GP, et al. A map of human genome variation from population scale sequencing. Nature. 2010;467(7319):1061.

[27] Belmont JW, Boudreau A, Leal SM, Hardenbol P, Pasternak S, Wheeler DA, et al. A haplotype map of the human genome. Nature. 2005;437(7063):1299–1320.

[28] Warner HR. THe role of computers in medical research. JAMA. 1966;196(11):944–949. Available from: `+http://dx.doi.org/10.1001/jama.1966.03100240078016`.

[29] Food, Administration D, et al. Innovation or stagnation: challenge and opportunity on the critical path to new medical products. Food and Drug Administration, critical path report. 2004;.

[30] Bickerstaffe A, Ranaweera T, Endersby T, Ellis C, Maddumarachchi S, Gooden GE, et al. The Ark: a customizable web-based data management tool for health and medical research. Bioinformatics. 2016;p. btw675.

[31] Harris PA, Taylor R, Thielke R, Payne J, Gonzalez N, Conde JG. Research electronic data capture (REDCap)-A metadata-driven methodology and workflow process for providing translational research informatics support. Journal of Biomedical Informatics. 2009;42(2):377–381. Cited By (since 1996):318.

[32] Nourie D. Java Technologies for Web Applications;. Accessed: 2017-09-30. http://www.oracle.com/technetwork/articles/java/webapps-1-138794.html.

[33] MySQL Documentation;. Accessed: 2018-04-15. https://dev.mysql.com/doc/.

[34] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Communications of the ACM. 2008;51(1):107–113.

[35] Graham RL, Shipman GM, Barrett BW, Castain RH, Bosilca G, Lumsdaine A. Open MPI: A high-performance, heterogeneous MPI; 2006. .

[36] Brodtkorb AR, Hagen TR, Sætra ML. GPU programming strategies and trends in GPU computing. Journal of Parallel and Distributed Computing. 2012;.

[37] Gaye A, Marcon Y, Isaeva J, LaFlamme P, Turner A, Jones EM, et al. DataSHIELD: taking the analysis to the data, not the data to the analysis. International journal of epidemiology. 2014;43(6):1929–1944.

[38] Yates A, Akanni W, Amode MR, Barrell D, Billis K, Carvalho-Silva D, et al. Ensembl 2016. Nucleic acids research. 2015;p. gkv1157.

[39] Speir ML, Zweig AS, Rosenbloom KR, Raney BJ, Paten B, Nejad P, et al. The UCSC genome browser database: 2016 update. Nucleic acids research. 2016;44(D1):D717–D725.

[40] Pedigree Drawing Software - Progeny Clinical;. Accessed: 2017-09-24. www.progenygenetics.com/clinical/pedigree.

[41] Zhao JH. Pedigree-drawing with R and graphviz. Bioinformatics. 2006;22(8):1013–1014.

[42] Tryka KA, Hao L, Sturcke A, Jin Y, Wang ZY, Ziyabari L, et al. NCBI's Database of Genotypes and Phenotypes: dbGaP. Nucleic acids research. 2013;42(D1):D975–D979.

[43] Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, et al. PLINK: A tool set for whole-genome association and population-based linkage analyses. American Journal of Human Genetics. 2007;81(3):559–575.

[44] DNA Sequencing Fact Sheet - National Human Genome Research Institute (NHGRI;. Accessed: 2017-09-25. https://www.genome.gov/10001177/dna-sequencing-fact-sheet/.

[45] Blankenberg D, Kuster GV, Coraor N, Ananda G, Lazarus R, Mangan M, et al. Galaxy: a web-based genome analysis tool for experimentalists. Current protocols in molecular biology. 2010;p. 19–10.

[46] Visscher PM, Brown MA, McCarthy MI, Yang J. Five years of GWAS discovery. American Journal of Human Genetics. 2012;90(1):7–24.

[47] Hamosh A, Scott AF, Amberger JS, Bocchini CA, McKusick VA. Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. Nucleic Acids Research. 2005;33(suppl 1):D514–D517.

[48] Genomic Data Is Going Google;. Accessed: 2018-04-20. https://cloud.google.com/genomics/resources/google-genomics-whitepaper.pdf.

[49] Newman S. Building microservices. " O'Reilly Media, Inc."; 2015.

[50] Kiran M, Murphy P, Monga I, Dugan J, Baveja SS. Lambda architecture for cost-effective batch and speed big data processing. In: Big Data (Big Data), 2015 IEEE International Conference on. IEEE; 2015. p. 2785–2792.

[51] Gray J. Data Management: Past, Present, and Future. arXiv preprint cs/0701156. 2007;.

[52] Jones DW. Punched cards-a brief illustrated technical history. Part of the punched card collection, The University of Iowa; 2005. Available from: http://homepage.cs.uiowa.edu/~jones/cards/history.html.

[53] Singh SK. Database systems: Concepts, design and applications. Pearson Education India; 2011.

[54] Imielinski T, Lipski Jr W. A systematic approach to relational database theory. In: Proceedings of the 1982 ACM SIGMOD international conference on Management of data. ACM; 1982. p. 8–14.

[55] Slazinski ED. In: Structured Query Language (SQL). John Wiley & Sons, Inc.; 2004. Available from: http://dx.doi.org/10.1002/047148296X.tie166.

[56] Bloor R. The Failure of Relational Data ba se, The Rise of Object Technology and the Need for the Hybrid Database; 2004.

[57] Martin RC. Agile software development: principles, patterns, and practices. Prentice Hall; 2002.

[58] Feuerlicht G. Database trends and directions: Current challenges and opportunities. vol. 567; 2010. p. 163–174.

[59] Wagner G. Object-Relational Databases. In: Foundations of Knowledge Systems. Springer; 1998. p. 71–83.

[60] Stonebraker M, Moore D. Object Relational DBMSs: The Next Great Wave. Morgan Kaufmann Publishers Inc.; 1995.

[61] Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, et al. Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems. 2008;26(2).

[62] Leavitt N. Will NoSQL Databases Live Up to Their Promise ? COMPUTER. 2010;43(2):12–14.

[63] Agrawal R, Ailamaki A, Bernstein PA, Brewer EA, Carey MJ, Chaudhuri S, et al. The Claremont report on database research. Communications of the ACM. 2009;52(6):56–65.

[64] Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, et al. A view of cloud computing. Communications of the ACM. 2010;53(4):50–58.

[65] Ward JS, Barker A. Undefined By Data: A Survey of Big Data Definitions. ArXiv e-prints. 2013 Sep;.

[66] Silberschatz A, Stonebraker M, Ullman J. Database Research: Achievements and Opportunities into the 21st Century. SIGMOD Record (ACM Special Interest Group on Management of Data). 1996;25(1):52–63.

[67] Indrawan-Santiago M. Database research: Are we at a crossroad? Reflection on NoSQL; 2012. p. 45–51.

[68] O'Driscoll A, Daugelaite J, Sleator RD. 'Big data', Hadoop and cloud computing in genomics. Journal of Biomedical Informatics. 2013;46(5):774–781.

[69] Ghemawat S, Gobioff H, Leung ST. The Google file system. In: ACM SIGOPS Operating Systems Review. vol. 37. ACM; 2003. p. 29–43.

[70] O'Connor B, Merriman B, Nelson S. SeqWare Query Engine: storing and searching sequence data in the cloud. BMC bioinformatics. 2010;11(Suppl 12):S2.

[71] Shvachko K, Kuang H, Radia S, Chansler R. The hadoop distributed file system. In: Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. IEEE; 2010. p. 1–10.

[72] Khetrapal A, Ganesh V. HBase and Hypertable for large scale distributed storage systems. Dept of Computer Science, Purdue University. 2006;.

[73] Kobler B, Berbert J, Caulk P, Hariharan PC. Architecture and design of storage and data management for the NASA Earth Observing System Data and Information System (EOSDIS); 1995. p. 65–76.

[74] Mailman MD, Feolo M, Jin Y, Kimura M, Tryka K, Bagoutdinov R, et al. The NCBI dbGaP database of genotypes and phenotypes. Nature Genetics. 2007;39(10):1181–1186.

[75] Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. Journal of molecular biology. 1990;215(3):403–410.

[76] Woodcock J, Woosley R. The FDA critical path initiative and its influence on new drug development [Review; Book Chapter]. ANNUAL REVIEW OF MEDICINE. 2008;59:1–12.

[77] Bernstam EV, Smith JW, Johnson TR. What is biomedical informatics? Journal of Biomedical Informatics. 2010;43(1):104 – 110.

[78] Sanders DL, Aronsky D. Biomedical Informatics Applications for Asthma Care: A Systematic Review. Journal of the American Medical Informatics Association. 2006;13(4):418–427.

[79] Saltz J, Oster S, Hastings S, Langella S, Kurc T, Sanchez W, et al. caGrid: Design and implementation of the core architecture of the cancer biomedical informatics grid. Bioinformatics. 2006;22(15):1910–1916.

[80] REDCap;. Accessed: 2018-04-05. https://www.project-redcap.org.

[81] Cavelaars M, Rousseau J, Parlayan C, de Ridder S, Verburg A, Ross R, et al. OpenClinica. Journal of clinical bioinformatics. 2015;5.

[82] Viangteeravat T, Brooks IM, Smith EJ, Furlotte N, Vuthipadadon S, Reynolds R, et al. Slim-prim: a biomedical informatics database to promote translational research. Perspectives in health information management / AHIMA, American Health Information Management Association. 2009;6:6.

[83] Wang X, Liu L, Fackenthal J, Cummings S, Olopade OI, Hope K, et al. Translational integrity and continuity: Personalized biomedical data integration. Journal of Biomedical Informatics. 2009;42(1):100–112.

[84] Mendel G. Experiments in plant hybridization (1865). Read at the February. 1996;8.

[85] Owen RD. Genetics in the 20th century. Journal of Heredity. 1983;74(5):314–319.

[86] Avery OT, MacLeod CM, McCarty M. Studies on the chemical nature of the substance inducing transformation of pneumococcal types induction of transformation by a desoxyribonucleic acid fraction isolated from pneumococcus type III. The Journal of experimental medicine. 1944;79(2):137–158.

[87] Nirenberg MW. The genetic code. Scientific American. 1963;208(3):80–95.

[88] Speyer JF, Lengyel P, Basilio C, Wahba AJ, Gardner RS, Ochoa S. Synthetic polynucleotides and the amino acid code. In: Cold Spring Harbor Symposia

on Quantitative Biology. vol. 28. Cold Spring Harbor Laboratory Press; 1963. p. 559–567.

[89] Jackson DA, Symons RH, Berg P. Biochemical method for inserting new genetic information into DNA of Simian Virus 40: circular SV40 DNA molecules containing lambda phage genes and the galactose operon of Escherichia coli. Proceedings of the National Academy of Sciences. 1972;69(10):2904–2909.

[90] Sanger F, Coulson AR. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. Journal of molecular biology. 1975;94(3):441–448.

[91] Langmead B, Trapnell C, Pop M, Salzberg SL, et al. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol. 2009;10(3):R25.

[92] Sanger F, Coulson AR, Hong G, Hill D, Petersen Gd. Nucleotide sequence of bacteriophage λ DNA. Journal of molecular biology. 1982;162(4):729–773.

[93] Botstein D, White RL, Skolnick M, Davis RW. Construction of a genetic linkage map in man using restriction fragment length polymorphisms. American journal of human genetics. 1980;32(3):314.

[94] Olson MV, Dutchik JE, Graham MY, Brodeur GM, Helms C, Frank M, et al. Random-clone strategy for genomic restriction mapping in yeast. Proceedings of the National Academy of Sciences. 1986;83(20):7826–7830.

[95] Coulson A, Sulston J, Brenner S, Karn J. Toward a physical map of the genome of the nematode Caenorhabditis elegans. Proceedings of the National Academy of Sciences. 1986;83(20):7821–7825.

[96] Putney SD, Herlihy WC, Schimmel P. A new troponin T and cDNA clones for 13 different muscle proteins, found by shotgun sequencing. Nature. 1983;302(5910):718–721.

[97] Lander ES, Linton LM, Birren B, Nusbaum C, Zody MC, Baldwin J, et al. Initial sequencing and analysis of the human genome. Nature. 2001;409(6822):860–921.

[98] Guttmacher AE, Collins FS. Genomic medicine - A primer. New England Journal of Medicine. 2002;347(19):1512–1520.

[99] Chakravarti A. Genomic contributions to Mendelian disease. Genome research. 2011;21(5):643–644.

[100] Botstein D, Risch N. Discovering genotypes underlying human phenotypes: Past successes for mendelian disease, future approaches for complex disease. Nature Genetics. 2003;33(SUPPL.):228–237.

[101] Walker FO. Huntington's disease. The Lancet. 2007;369(9557):218 – 228.

[102] Teare MD, Barrett JH. Genetic linkage studies. The Lancet. 2005;366(9490):1036 – 1044. Available from: `http://www.sciencedirect.com/science/article/pii/S0140673605673825`.

[103] Riordan JR, Rommens JM, Kerem BS, Alon N, Rozmahel R, Grzelczak Z, et al. Identification of the cystic fibrosis gene: Cloning and characterization of complementary DNA. Science. 1989;245(4922):1066–1073.

[104] Morton NE. Significance levels in complex inheritance. The American Journal of Human Genetics. 1998;62(3):690–697.

[105] Cordell HJ, Clayton DG. Genetic association studies. The Lancet. 2005;366(9491):1121–1131.

[106] Visscher PM, Montgomery GW. Genome-wide association studies and human disease. JAMA: The Journal of the American Medical Association. 2009;302(18):2028–2029.

[107] Ha NT, Freytag S, Bickeboeller H. Coverage and efficiency in current SNP chips. European Journal of Human Genetics. 2014;22(9):1124–1130.

[108] Anderson CA, Pettersson FH, Barrett JC, Zhuang JJ, Ragoussis J, Cardon LR, et al. Evaluating the Effects of Imputation on the Power, Coverage, and Cost Efficiency of Genome-wide SNP Platforms. The American Journal of Human Genetics. 2008;83(1):112 – 119.

[109] Cantor RM, Lange K, Sinsheimer JS. Prioritizing GWAS results: a review of statistical methods and recommendations for their application. The American Journal of Human Genetics. 2010;86(1):6–22.

[110] Laurie CC, Doheny KF, Mirel DB, Pugh EW, Bierut LJ, Bhangale T, et al. Quality control and quality assurance in genotypic data for genome-wide association studies. Genetic Epidemiology. 2010;34(6):591–602.

[111] Bush WS, Moore JH. Chapter 11: Genome-Wide Association Studies. PLoS Comput Biol. 2012 12;8(12):e1002822.

[112] Marchini J, Donnelly P, Cardon LR. Genome-wide strategies for detecting multiple loci that influence complex diseases. Nature Genetics. 2005;37(4):413–417.

[113] Bush WS, Moore JH. Genome-wide association studies. PLoS computational biology. 2012;8(12):e1002822.

[114] Goecks J, Nekrutenko A, Taylor J. Lessons learned from galaxy, a web-based platform for high-throughput genomic analyses. In: E-Science (e-Science), 2012 IEEE 8th International Conference on. IEEE; 2012. p. 1–6.

[115] Muñiz-Fernandez F, Carreño-Torres A, Morcillo-Suarez C, Navarro A. Genome-wide association studies pipeline (GWASpi): a desktop application for genome-wide SNP analysis and management. Bioinformatics. 2011;27(13):1871–1872.

[116] Panagiotou OA, Willer CJ, Hirschhorn JN, Ioannidis JP. The power of meta-analysis in genome-wide association studies. Annual review of genomics and human genetics. 2013;14:441–465.

[117] DeWan A, Liu M, Hartman S, Zhang SSM, Liu DTL, Zhao C, et al. HTRA1 promoter polymorphism in wet age-related macular degeneration. Science. 2006;314(5801):989–992.

[118] Wang G. Chromosome 10q26 locus and age-related macular degeneration: A progress update. Experimental Eye Research. 2014;119:1–7.

[119] Burton PR, Clayton DG, Cardon LR, Craddock N, Deloukas P, Duncanson A, et al. Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls. Nature. 2007;447(7145):661–678.

[120] Easton DF, Eeles RA. Genome-wide association studies in cancer. Human Molecular Genetics. 2008;17(R2):R109–R115.

[121] Han MR, Long J, Choi JY, Low SK, Kweon SS, Zheng Y, et al. Genome-wide association study in East Asians identifies two novel breast cancer susceptibility loci. Human Molecular Genetics. 2016;25(15):3361–3371.

[122] Palomba G, Loi A, Porcu E, Cossu A, Zara I, Budroni M, et al. Genome-wide association study of susceptibility loci for breast cancer in Sardinian population. BMC cancer. 2015;15(1):383.

[123] Haryono SJ, Datasena I, Santosa WB, Mulyarahardja R, Sari K. A pilot genome-wide association study of breast cancer susceptibility loci in Indonesia. Asian Pacific Journal of Cancer Prevention. 2015;16(6):2231–2235.

[124] Fejerman L, Ahmadiyeh N, Hu D, Huntsman S, Beckman KB, Caswell JL, et al. Genome-wide association study of breast cancer in Latinas identifies novel protective variants on 6q25. Nature communications. 2014;5.

[125] Fletcher O, Johnson N, Orr N, Hosking FJ, Gibson LJ, Walker K, et al. Novel breast cancer susceptibility locus at 9q31. 2: results of a genome-wide association study. Journal of the National Cancer Institute. 2011;.

[126] Cai Q, Zhang B, Sung H, Low SK, Kweon SS, Lu W, et al. Genome-wide association analysis in East Asians identifies breast cancer susceptibility loci at 1q32. 1, 5q14. 3 and 15q26. 1. Nature genetics. 2014;46(8):886–890.

[127] Low SK, Takahashi A, Ashikawa K, Inazawa J, Miki Y, Kubo M, et al. Genome-wide association study of breast cancer in the Japanese population. PloS one. 2013;8(10):e76463.

[128] Richardson SS, Stevens H. Postgenomics: Perspectives on biology after the genome. Duke University Press; 2015.

[129] Lango Allen H, Estrada K, Lettre G, Berndt SI, Weedon MN, Rivadeneira F, et al. Hundreds of variants clustered in genomic loci and biological pathways affect human height. Nature. 2010;467(7317):832–8.

[130] Carr G. Biology 2.0: A special report on the human genome. Economist Newspaper; 2010.

[131] Council NR, et al. Toward precision medicine: building a knowledge network for biomedical research and a new taxonomy of disease. National Academies Press; 2011.

[132] Hunter DJ. Uncertainty in the era of precision medicine. New England Journal of Medicine. 2016;375(8):711–713.

[133] Kubo M. BioBank Japan project: Epidemiological study. Journal of Epidemiology. 2017;p. –. Available from: `http://www.sciencedirect.com/science/article/pii/S0917504016301447`.

[134] Feng Wc, Feng X, Ge R. Green supercomputing comes of age. IT professional. 2008;10(1):17–23.

[135] Hey AJG. High-performance computing - Past, present and future. Computing and Control Engineering Journal. 1997;8(1):33–42.

[136] Dongarra J, Sterling T, Simon H, Strohmaier E. High-performance computing: Clusters, constellations, MPPs, and future directions. Computing in Science and Engineering. 2005;7(2):51–59.

[137] Nodes, Sockets, Cores and FLOPS, Oh, My;. Accessed: 2014-02-20. http://en.community.dell.com/techcenter/high-performance-computing/w/wiki/2329.aspx.

[138] Fosdick LD. An introduction to high-performance scientific computing. MIT Press; 1996.

[139] Top 500, November 2017 ;. Accessed: 2018-04-05. `https://www.top500.org/lists/2017/11/`.

[140] Singh HJ, Singh V. High Scalability of HDFS using Distributed Name space. International Journal of Computer Applications. 2012;52.

[141] Graham RL, Barrett BW, Shipman GM, Woodall TS, Bosilca G. Open MPI: A high performance, flexible implementation of MPI point-to-point communications. Parallel Processing Letters. 2007;17(1):79–88.

[142] Gabriel E, Fagg GE, Bosilca G, Angskun T, Dongarra JJ, Squyres JM, et al. Open MPI: Goals, concept, and design of a next generation MPI implementation. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2004;3241:97–104.

[143] Dagum L, Menon R. OpenMP: an industry standard API for shared-memory programming. Computational Science & Engineering, IEEE. 1998;5(1):46–55.

[144] Bell G, Gray J. What's next in high-performance computing? Communications of the ACM. 2002;45(2):91–95.

[145] Ceruzzi PE. A history of modern computing. MIT press; 2003.

[146] Reilly ED. Milestones in computer science and information technology. Greenwood Publishing Group; 2003.

[147] Graham SL, Snir M, Patterson CA, et al. Getting up to speed: The future of supercomputing. National Academies Press; 2005.

[148] Becker DJ, Sterling T, Savarese D, Dorband JE, Ranawak UA, Packer CV. BEOWULF: A parallel workstation for scientific computation. In: Proceedings, International Conference on Parallel Processing. vol. 95; 1995. .

[149] Huang S, Xiao S, Feng Wc. On the energy efficiency of graphics processing units for scientific computing. In: Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on. IEEE; 2009. p. 1–8.

[150] Foster I, Kesselman C. The Grid 2: Blueprint for a new computing infrastructure. Elsevier; 2003.

[151] Shiers J. The Worldwide LHC Computing Grid (worldwide LCG). Computer Physics Communications. 2007;177(1-2 SPEC. ISS.):219–223. Cited By (since 1996)22. Available from: http://www.scopus.com/inward/record.url?eid=2-s2.0-34250653676&partnerID=40&md5=02c766929c971500588193bc3d78f81c.

[152] Foster I, Zhao Y, Raicu I, Lu S. Cloud Computing and Grid Computing 360-degree compared; 2008. .

[153] Sundararajan P. High performance computing using FPGAs. Xilinx White Paper: FPGAs. 2010;p. 1–15.

[154] Geist A, Reed DA. A survey of high-performance computing scaling challenges. The International Journal of High Performance Computing Applications. 2017;31(1):104–113.

[155] Thomas DP. Sailors, scurvy and science. Journal of the Royal Society of Medicine. 1997;90(1):50.

[156] Paneth N, Vinten-Johansen P, Brody H, Rip M. A rivalry of foulness: official and unofficial investigations of the London cholera epidemic of 1854. American Journal of Public Health. 1998;88(10):1545–1553.

[157] Wojciechowski T, Sakowicz B, Makowski D, Napieralski A. Transaction system with reporting capability in a web-based data warehouse application developed in Oracle Application Express. In: CAD Systems in Microelectronics, 2009. CADSM 2009. 10th International Conference-The Experience of Designing and Application of. IEEE; 2009. p. 273–276.

[158] Parnas DL. On the criteria to be used in decomposing systems into modules. Communications of the ACM. 1972;15(12):1053–1058.

[159] Denning DE. An intrusion-detection model. IEEE Transactions on software engineering. 1987;(2):222–232.

[160] Aziz RA, Wong B. The Interplay between Requirements Relationships Knowledge and Requirements Change towards Software Project Success: An Assessment Using Partial Least Square (PLS). Procedia Computer Science. 2015;46:732–741.

[161] Moran A. Agile Project Management. In: Managing Agile. Springer; 2015. p. 71–101.

[162] Dybå T, Dingsøyr T. Empirical studies of agile software development: A systematic review. Information and software technology. 2008;50(9):833–859.

[163] Gajowniczek P, Zawiślak R. Cross team, multi task, multi flow kanban application. Studia Informatica. 2015;36(1):19–30.

[164] Davis W, Knuiman M, Davis T. An Australian cardiovascular risk equation for type 2 diabetes: the Fremantle Diabetes Study. Internal medicine journal. 2010;40(4):286–292.

[165] McKnight CM, Newnham JP, Stanley FJ, Mountain JA, Landau LI, Beilin LJ, et al. Birth of a cohort—the first 20 years of the Raine study. Medical Journal of Australia. 2012;197(11):608.

[166] Deroach J, Mclaren T, Paterson R, Hoffmann L, Hewitt A, Mackey D, et al. 9. The Australian Inherited Retinal Disease Register And Dna Bank Part 1-Structure And Current Status. Clinical & Experimental Ophthalmology. 2012;40:39.

[167] Yeong W, Howes T, Kille S. Lightweight directory access protocol; 1995.

[168] Ranaweera T, Makalic E, Hopper JL, Bickerstaffe A. An open-source, integrated pedigree data management and visualization tool for genetic epidemiology. International journal of epidemiology. 2018;.

[169] for Disease Control C, (CDC P, et al. Awareness of family health history as a risk factor for disease–United States, 2004. MMWR Morbidity and mortality weekly report. 2004;53(44):1044.

[170] Doerr M, Teng K. Family history: still relevant in the genomics era. Cleveland Clinic journal of medicine. 2012;79(5):331–336.

[171] Bennett RL, Steinhaus KA, Uhrich SB, O'Sullivan CK, Resta RG. Recommndation for standardized human pedigree nomenclature. American Journal of Human Genetics. 1995;56(3):745–752.

[172] Tuttle C, Nonato LG, Silva CT. PedVis: A Structured, Space-Efficient Technique for Pedigree Visualization. Visualization and Computer Graphics, IEEE Transactions on. 2010 Nov;16(6):1063–1072.

[173] Santos JM, Sousa Santos B, Dias P, Silva S, Ferreira C. Extending the H-Tree Layout Pedigree: An Evaluation. In: Information Visualisation (IV), 2013 17th International Conference; 2013. p. 422–427.

[174] Sallaberry A, Fu Y, Ho H, Ma K. ContactTrees: A Technique for Studying Personal Network Data. CoRR. 2014;abs/1411.0052.

[175] Cohen LD, Cohen I. Finite-element methods for active contour models and balloons for 2-D and 3-D images. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 1993;15(11):1131–1147.

[176] Brun-Samarcq L, Gallina S, Philippi A, Demenais F, Vaysseix G, Barillot E. CoPE: a collaborative pedigree drawing environment. Bioinformatics. 1999;15(4):345–346.

[177] Loh AM, Wiltshire S, Emery J, Carter KW, Palmer LJ. Celestial3D: a novel method for 3D visualization of familial data. Bioinformatics. 2008;24(9):1210–1211.

[178] Paterson T, Graham M, Kennedy J, Law A. VIPER: a visualisation tool for exploring inheritance inconsistencies in genotyped pedigrees. BMC Bioinformatics. 2012;13(Supplment 8):1 – 16.

[179] Sinnwell JP, Therneau TM, Schaid DJ. The kinship2 R package for pedigree data. Human heredity. 2014;78(2):91–93.

[180] Voorrips RE, Bink MC, van de Weg WE. Pedimap: software for the visualization of genetic and phenotypic data in pedigrees. Journal of Heredity. 2012;p. ess060.

[181] Trager EH, Khanna R, Marrs A, Siden L. Madeline 2.0 PDE: a new program for local and web-based pedigree drawing. Bioinformatics. 2007;23(14):1854–1856.

[182] Knuth DE. Fundamental algorithms: the art of computer programming; 1973.

[183] Nguyen TL, Schmidt DF, Makalic E, Dite GS, Stone J, Apicella C, et al. Explaining Variance in the Cumulus Mammographic Measures That Predict Breast Cancer Risk: A Twins and Sisters Study. Cancer Epidemiology Biomarkers & Prevention. 2013;22(12):2395–2403. Twins and sisters data-set.

[184] Hamamy H. Consanguineous marriages. Journal of community genetics. 2012;3(3):185–192.

[185] Bass L. Software architecture in practice. Pearson Education India; 2012.

[186] Taylor RN, Medvidovic N, Dashofy EM. Software architecture: foundations, theory, and practice. Wiley Publishing; 2009.

[187] Shaw M, Clements P. The golden age of software architecture. IEEE software. 2006;23(2):31–39.

[188] Aoyama M, et al. New age of software development: How component-based software engineering changes the way of software development. In: 1998 International Workshop on CBSE. Citeseer; 1998. p. 1–5.

[189] Manuel PD, AlGhamdi J. A data-centric design for n-tier architecture. Information Sciences. 2003;150(3):195–206.

[190] Shan TC, Hua WW. Solution architecture for n-tier applications. In: Services Computing, 2006. SCC'06. IEEE International Conference on. IEEE; 2006. p. 349–356.

[191] Zhang LJ, Zhang J, Cai H. Service-oriented architecture. Services Computing. 2007;p. 89–113.

[192] Krishnamurty B, Rexford J. Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.; 2001.

[193] Laskey KB, Laskey K. Service oriented architecture. Wiley Interdisciplinary Reviews: Computational Statistics. 2009;1(1):101–105.

[194] Thönes J. Microservices. IEEE Software. 2015;32(1):116–116.

[195] Kosar T. Data Intensive Distributed Computing: Challenges and Solutions for Large-scale Information Management: Challenges and Solutions for Large-scale Information Management. IGI Global; 2012.

[196] Di Francesco P, Lago P, Malavolta I. Research on Architecting Microservices: Trends, Focus, and Potential for Industrial Adoption. In: Proceedings of the 14th International Conference on Software Architecture (ICSA); 2017. .

[197] Manteli C, Van Den Hooff B, Tang A, Van Vliet H. The impact of multi-site software governance on knowledge management. In: Global Software Engineering (ICGSE), 2011 6th IEEE International Conference on. IEEE; 2011. p. 40–49.

[198] Marru S, Pierce M, Pamidighantam S, Wimalasena C. Apache Airavata as a laboratory: architecture and case study for component-based gateway middleware. In: Proceedings of the 1st Workshop on The Science of Cyberinfrastructure: Research, Experience, Applications and Models. ACM; 2015. p. 19–26.

[199] Reed E, Nunez S, Kulp D, Qian J, Reilly MP, Foulkes AS. A guide to genome-wide association analysis and post-analytic interrogation. Statistics in medicine. 2015;34(28):3769–3792.

[200] Dowd M, McDonald J, Schuh J. The art of software security assessment: Identifying and preventing software vulnerabilities. Pearson Education; 2006.

[201] Anderson RJ. Security engineering: a guide to building dependable distributed systems. John Wiley & Sons; 2010.

[202] Scott J, Kazman R. Realizing and refining architectural tactics: Availability. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST; 2009.

[203] Kazman R, Gagliardi M, Wood W. Scaling up software architecture analysis. Journal of Systems and Software. 2012;85(7):1511–1519.

[204] Brownsword LL, Carney DJ, Fisher D, Lewis G, Meyers C. Current perspectives on interoperability. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST; 2004.

[205] Curbera F, Duftler M, Khalaf R, Nagy W, Mukhi N, Weerawarana S. Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. IEEE Internet computing. 2002;6(2):86–93.

[206] Bachmann F, Bass L, Nord R. Modifiability tactics. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST; 2007.

[207] Bachmann F, Bass L, Klein M, Shelton C. Experience using an expert system to assist an architect in designing for modifiability. In: Software Architecture, 2004. WICSA 2004. Proceedings. Fourth Working IEEE/IFIP Conference on. IEEE; 2004. p. 281–284.

[208] Liu HH. Software performance and scalability: a quantitative approach. vol. 7. John Wiley & Sons; 2011.

[209] Freeman S, Pryce N. Growing Object-oriented Software: Guided by Tests. Pearson Education India; 2009.

[210] Bass L, John BE. Linking usability to software architecture patterns through general scenarios. Journal of Systems and Software. 2003;66(3):187 – 197. Software architecture – Engineering quality attributes. Available from: http://www.sciencedirect.com/science/article/pii/S0164121202000766.

[211] Barrett T, Troup DB, Wilhite SE, Ledoux P, Evangelista C, Kim IF, et al. NCBI GEO: archive for functional genomics data sets—10 years on. Nucleic acids research. 2011;39(suppl 1):D1005–D1010.

[212] Mitha F, Herodotou H, Borisov N, Jiang C, Yoder J, Owzar K. SNPpy-database management for SNP data from Genome wide association studies. PloS one. 2011;6(10):e24982.

[213] Turner S, Armstrong LL, Bradford Y, Carlson CS, Crawford DC, Crenshaw AT, et al. Quality control procedures for genome-wide association studies. Current protocols in human genetics. 2011;p. 1–19.

[214] Stonebraker M. SQL databases v. NoSQL databases. Communications of the ACM. 2010;53(4):10–11.

[215] Cock PJ, Fields CJ, Goto N, Heuer ML, Rice PM. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. Nucleic acids research. 2010;38(6):1767–1771.

[216] Bernroider E, Koch S. ERP selection process in midsize and large organizations. Business Process Management Journal. 2001;7(3):251–257.

[217] Ngai EW, Law CC, Wat FK. Examining the critical success factors in the adoption of enterprise resource planning. Computers in industry. 2008;59(6):548–564.

[218] Aloini D, Dulmin R, Mininno V. Risk management in ERP project introduction: Review of the literature. Information & Management. 2007;44(6):547–567.

[219] Shehab E, Sharp M, Supramaniam L, Spedding TA. Enterprise resource planning: An integrative review. Business Process Management Journal. 2004;10(4):359–386.

[220] Keil M, Tiwana A. Relative importance of evaluation criteria for enterprise systems: a conjoint study. Information Systems Journal. 2006;16(3):237–262.

[221] Benlian A, Hess T. Comparing the relative importance of evaluation criteria in proprietary and open-source enterprise application software selection–a conjoint study of ERP and Office systems. Information Systems Journal. 2011;21(6):503–525.

[222] An extensible cross-language static code analyzer.);. Accessed: 2019-04-10. `https://pmd.github.io/`.

[223] Louridas P. Static code analysis. IEEE Software. 2006;23(4):58–61.

[224] Java Rules);. Accessed: 2019-04-10. `https://pmd.github.io/latest/pmd_rules_java.html#documentation`.

[225] Brooke J. SUS-A quick and dirty usability scale. Usability evaluation in industry. 1996;189(194):4–7.

[226] Boone HN, Boone DA. Analyzing likert data. Journal of extension. 2012;50(2):1–5.

[227] Michailidou K, Hall P, Gonzalez-Neira A, Ghoussaini M, Dennis J, Milne RL, et al. Large-scale genotyping identifies 41 new loci associated with breast cancer risk. Nature genetics. 2013;45(4):353–361.

[228] Venter J, Adams M, Myers E, Li P, Mural R, Sutton G, et al. The sequence of the human genome [Review]. SCIENCE. 2001 FEB 16;291(5507):1304+.

[229] Wang G, Tang J. The nosql principles and basic application of cassandra model. In: Computer Science & Service System (CSSS), 2012 International Conference on. IEEE; 2012. p. 1332–1335.

[230] Bycroft C, Freeman C, Petkova D, Band G, Elliott LT, Sharp K, et al. Genome-wide genetic data on 500,000 UK Biobank participants. bioRxiv. 2017;Available from: https://www.biorxiv.org/content/early/2017/07/20/166298.

[231] Ott J, Wang J, Leal SM. Genetic linkage analysis in the age of whole-genome sequencing. Nature Reviews Genetics. 2015;16(5):275–284.

[232] de Paula R, Holanda M, Gomes LS, Lifschitz S, Walter MEM. Provenance in bioinformatics workflows. BMC bioinformatics. 2013;14(11):S6.

[233] Thorvaldsdóttir H, Robinson JT, Mesirov JP. Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. Briefings in bioinformatics. 2013;14(2):178–192.

[234] Krzywinski M, Schein J, Birol I, Connors J, Gascoyne R, Horsman D, et al. Circos: an information aesthetic for comparative genomics. Genome research. 2009;19(9):1639–1645.